Panduan Developer, Versi 2

AWS IoT Greengrass

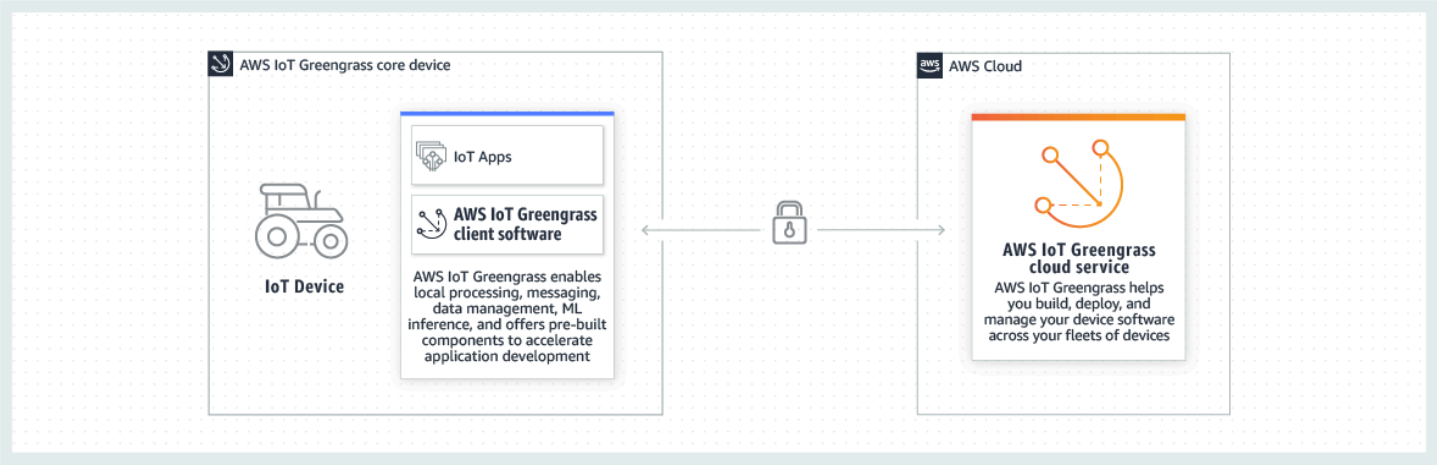
AWS IoT Greengrass: Panduan Developer, Versi 2

Table of Contents

Apakah AWS IoT Greengrass itu?

AWS IoT Greengrass adalah waktu aktif edge Internet untuk Segala (IoT) sumber terbuka dan layanan cloud yang membantu Anda membangun, men-deploy, dan mengelola aplikasi IoT pada perangkat Anda. Anda dapat menggunakan AWS IoT Greengrass untuk membangun perangkat lunak yang memungkinkan perangkat Anda untuk bertindak secara lokal pada data yang dihasilkannya, menjalankan prediksi berdasarkan model machine learning, serta memfilter dan mengumpulkan data perangkat. AWS IoT Greengrass memungkinkan perangkat Anda untuk mengumpulkan dan menganalisis data lebih dekat ke tempat data tersebut dihasilkan, bereaksi secara otonom terhadap peristiwa lokal, dan berkomunikasi secara aman dengan perangkat lain di jaringan lokal. Perangkat Greengrass juga dapat berkomunikasi dengan aman dengan AWS IoT Core dan mengekspor data IoT ke AWS Cloud. Anda dapat menggunakan AWS IoT Greengrass untuk membangun aplikasi edge dengan menggunakan modul perangkat lunak bawaan, yang disebut komponen, yang dapat menghubungkan perangkat edge Anda ke layanan AWS atau layanan pihak ketiga. Anda juga dapat menggunakan AWS IoT Greengrass untuk mengemas dan menjalankan perangkat lunak Anda dengan menggunakan fungsi Lambda, kontainer Docker, proses sistem operasi asli, atau waktu aktif kustom pilihan Anda.

Contoh berikut menunjukkan bagaimana perangkat AWS IoT Greengrass berinteraksi dengan AWS Cloud.



Fitur baru

AWS IoT Greengrass V2memperkenalkan fitur dan peningkatan baru. Berikut ini mencakup informasi lebih lanjut tentang fitur-fitur baru yang ditawarkan di versi 2.

- Apa yang baru di AWS IoT Greengrass Version 2

Untuk pengguna AWS IoT Greengrass pertama kali

Jika Anda baru menggunakan AWS IoT Greengrass, kami menyarankan agar Anda membaca bagian berikut ini:

- Cara kerja AWS IoT Greengrass

Selanjutnya, ikuti Tutorial memulai untuk mencoba fitur dasar AWS IoT Greengrass. Dalam tutorial ini, Anda akan menginstal perangkat lunak inti AWS IoT Greengrass pada perangkat, mengembangkan komponen Hello World, dan mengemas komponen untuk deployment.

Untuk pengguna AWS IoT Greengrass yang sudah ada

Untuk pengguna saat iniAWS IoT Greengrass V1, kami merekomendasikan topik berikut untuk membantu Anda memahami perbedaan antara Greengrass versi 1 dan Greengrass versi 2, dan mempelajari cara beralih dari versi 1 ke versi 2:

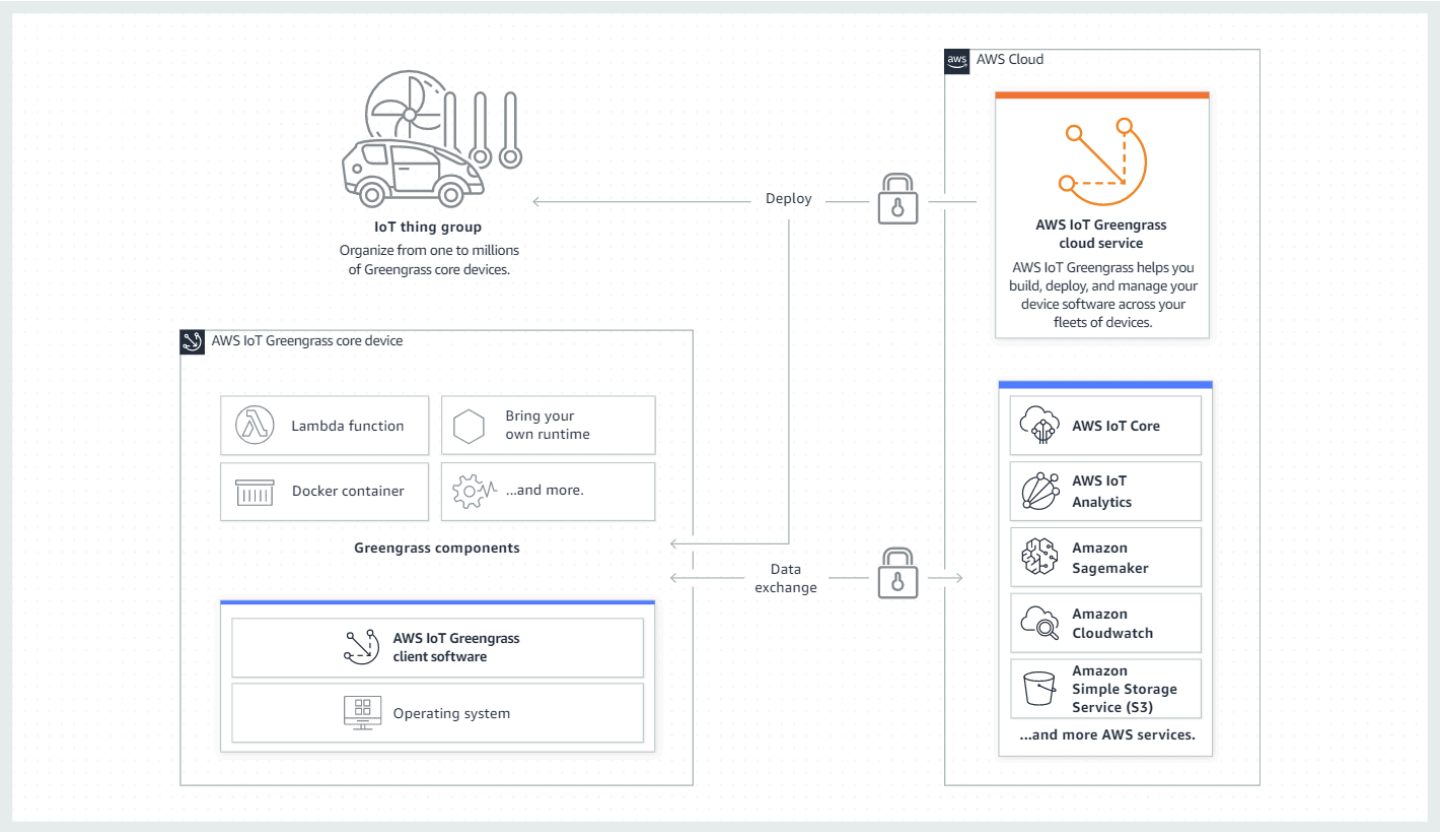
- Migrasi dari AWS IoT Greengrass Versi 1

Cara kerja AWS IoT Greengrass

Perangkat lunak AWS IoT Greengrass klien, juga disebut perangkat lunak AWS IoT Greengrass Core, berjalan pada distribusi berbasis Windows dan Linux, seperti Ubuntu atau Raspberry Pi OS, untuk perangkat dengan arsitektur ARM atau x86. Dengan AWS IoT Greengrass, Anda dapat memprogram perangkat untuk bertindak secara lokal pada data yang dihasilkan, menjalankan prediksi berdasarkan model machine learning, serta memfilter dan mengumpulkan data perangkat. AWS IoT Greengrass memungkinkan eksekusi lokal dari fungsi AWS Lambda, kontainer Docker, proses OS asli, atau waktu aktif kustom pilihan Anda.

AWS IoT Greengrass menyediakan modul perangkat lunak bawaan yang disebut komponen yang memungkinkan Anda dengan mudah memperluas fungsionalitas perangkat edge. Komponen AWS IoT Greengrass memungkinkan Anda untuk terhubung ke layanan AWS dan aplikasi pihak ketiga di edge tersebut. Setelah Anda mengembangkan aplikasi IoT Anda, AWS IoT Greengrass memungkinkan Anda untuk menggunakan, mengonfigurasi, dan mengelola aplikasi tersebut dari jarak jauh di armada perangkat Anda di lapangan.

Contoh berikut menunjukkan cara perangkat AWS IoT Greengrass berinteraksi dengan layanan cloud AWS IoT Greengrass dan layanan AWS lainnya di AWS Cloud.



Konsep kunci untuk AWS IoT Greengrass

Berikut ini adalah konsep penting untuk memahami dan menggunakan AWS IoT Greengrass:

AWS IoThal

AWS IoTSesuatu adalah representasi dari perangkat tertentu atau entitas logis. Informasi tentang suatu hal disimpan dalam AWS IoT registri.

Perangkat inti Greengrass

Perangkat yang menjalankan perangkat lunak inti AWS IoT Greengrass. Perangkat inti Greengrass adalah sebuah objek IoT AWS. Anda dapat menambahkan beberapa perangkat inti ke grup objek AWS IoT untuk membuat dan mengelola grup perangkat inti Greengrass. Untuk informasi selengkapnya, lihat Menyiapkan perangkat AWS IoT Greengrass inti.

Perangkat klien Greengrass

Perangkat yang terhubung ke dan berkomunikasi dengan perangkat inti Greengrass melalui MQTT. Perangkat klien Greengrass adalah sebuah objek AWS IoT. Perangkat inti dapat memproses, memfilter, dan mengumpulkan data dari perangkat klien yang terhubung dengannya.

Anda dapat mengonfigurasi perangkat inti untuk merelai pesan MQTT antara perangkat klien, layanan cloud AWS IoT Core, dan komponen Greengrass. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Perangkat klien dapat menjalankan FreeRTOS atau menggunakan AWS IoT Device SDK atau API penemuan Greengrass untuk mendapatkan informasi tentang perangkat inti yang dapat dihubungkannya.

Komponen Greengrass

Sebuah modul perangkat lunak yang di-deploy ke dan berjalan pada perangkat inti Greengrass. Semua perangkat lunak yang dikembangkan dan digunakan dengan AWS IoT Greengrass dimodelkan sebagai sebuah komponen. AWS IoT Greengrass menyediakan komponen publik bawaan yang menyediakan fitur dan fungsionalitas yang dapat Anda gunakan dalam aplikasi Anda. Anda juga dapat mengembangkan komponen kustom Anda sendiri, di perangkat lokal Anda atau di cloud. Setelah Anda mengembangkan komponen kustom, Anda dapat menggunakan layanan cloud AWS IoT Greengrass untuk men-deploy komponen itu ke perangkat inti tunggal atau ganda. Anda dapat membuat komponen kustom dan men-deploy komponen tersebut ke perangkat inti. Ketika Anda melakukannya, perangkat inti akan mengunduh sumber daya berikut untuk menjalankan komponen tersebut:

- Resep: Sebuah file JSON atau YAML yang menjelaskan modul perangkat lunak dengan menentukan detail, konfigurasi, dan parameter komponen.
- Artifact: Kode sumber, biner, atau skrip yang menentukan perangkat lunak yang akan berjalan pada perangkat Anda. Anda dapat membuat artefak dari nol, atau Anda dapat membuat komponen dengan menggunakan fungsi Lambda, kontainer Docker, atau waktu aktif kustom.
- Dependensi: Hubungan antara komponen yang memungkinkan Anda untuk menerapkan pembaruan otomatis atau restart komponen dependen. Misalnya, Anda dapat memiliki komponen pemrosesan pesan aman yang tergantung pada komponen enkripsi. Hal ini memastikan bahwa setiap pembaruan untuk komponen enkripsi secara otomatis memperbarui dan me-restart komponen pemrosesan pesan.

Lihat informasi yang lebih lengkap di Komponen yang disediakan oleh AWS dan Kembangkan AWS IoT Greengrass komponen.

Deployment

Proses untuk mengirim komponen dan menerapkan konfigurasi komponen yang diinginkan pada perangkat target tujuan, yang dapat menjadi perangkat inti Greengrass tunggal atau grup perangkat inti Greengrass. Deployment secara otomatis menerapkan konfigurasi komponen

yang diperbarui ke target dan mencakup komponen lain yang didefinisikan sebagai dependensi. Anda juga dapat mengkloning deployment yang ada untuk membuat deployment baru yang menggunakan komponen yang sama tetapi di-deploy pada target yang berbeda. Deployment bersifat terus menerus, yang berarti bahwa setiap update yang Anda buat untuk komponen atau konfigurasi komponen deployment akan secara otomatis dikirim ke semua target tujuan. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat.

AWS IoT GreengrassPerangkat lunak inti

Serangkaian seluruh perangkat lunak AWS IoT Greengrass yang Anda instal pada perangkat inti. AWS IoT Greengrass Perangkat lunak inti terdiri dari berikut ini:

- Nukleus: Komponen yang diperlukan ini menyediakan fungsionalitas minimum perangkat lunak inti AWS IoT Greengrass. Nukleus ini mengelola deployment, orkestrasi, dan manajemen siklus hidup komponen lainnya. Ia juga memfasilitasi komunikasi antar komponen AWS IoT Greengrass secara lokal pada perangkat individual. Untuk informasi selengkapnya, lihat Inti Greengrass.
- Komponen opsional: Komponen yang dapat dikonfigurasi ini disediakan oleh AWS IoT Greengrass dan mengaktifkan fitur tambahan di perangkat edge Anda. Tergantung pada kebutuhan Anda, Anda dapat memilih komponen opsional yang ingin Anda deploy ke perangkat Anda, seperti streaming data, inferensi machine learning lokal, atau antarmuka baris perintah lokal. Untuk informasi selengkapnya, lihat Komponen yang disediakan oleh AWS.

Anda dapat memperbarui perangkat lunak inti AWS IoT Greengrass dengan menerapkan versi baru komponen Anda ke perangkat Anda.

Fitur dari AWS IoT Greengrass

AWS IoT Greengrass Version 2 terdiri atas elemen-elemen berikut:

- Distribusi perangkat lunak
  - Komponen inti Greengrass, yang merupakan instalasi minimum perangkat lunak Core. AWS IoT Greengrass Komponen ini mengelola deployment, orkestrasi, dan manajemen siklus hidup komponen Greengrass.
  - Komponen tambahan AWS yang disediakan opsional yang terintegrasi dengan layanan, protokol, dan perangkat lunak.
  - Alat pengembangan Greengrass, yang dapat Anda gunakan untuk membuat, menguji, membangun, menerbitkan, dan menyebarkan komponen Greengrass kustom.

  - TheAWS IoT Device SDK, yang berisi pustaka komunikasi antarproses (IPC) untuk komponen Greengrass kustom dan pustaka penemuan Greengrass untuk perangkat klien.
  - Stream Manager SDK, yang dapat Anda gunakan untuk mengelola aliran data pada perangkat inti.
- Layanan cloud
  - API AWS IoT Greengrass V2
  - Konsol AWS IoT Greengrass V2

Perangkat lunak inti AWS IoT Greengrass

Anda dapat menggunakan perangkat lunak inti AWS IoT Greengrass yang berjalan pada perangkat edge Anda untuk melakukan hal berikut:

- Memproses aliran data pada perangkat lokal dengan ekspor otomatis ke cloud AWS. Untuk informasi selengkapnya, lihat Kelola aliran data di perangkat inti Greengrass.
- Mendukung olahpesan MQTT antara AWS IoT dan komponen. Untuk informasi selengkapnya, lihat Terbitkan/berlangganan pesan MQTT AWS IoT Core.
- Berinteraksi dengan perangkat lokal yang terhubung dan berkomunikasi melalui MQTT. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.
- Mendukung olahpesan publikasi dan berlangganan lokal antar komponen. Untuk informasi selengkapnya, lihat Pesan lokal publikasi/berlangganan.
- Men-deploy dan memanggil komponen dan fungsi Lambda. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat.
- Mengelola siklus hidup komponen, seperti dengan dukungan untuk menginstal dan menjalankan skrip. Untuk informasi selengkapnya, lihat AWS IoT Greengrass referensi resep komponen.
- Lakukan pembaruan perangkat lunak over-the-air (OTA) yang aman dari perangkat lunak AWS IoT Greengrass Core dan komponen khusus. Lihat informasi yang lebih lengkap di Perbarui perangkat lunak inti AWS IoT Greengrass (OTA) dan Deploy komponen AWS IoT Greengrass ke perangkat.
- Menyediakan penyimpanan rahasia lokal yang aman dan terenkripsi dan akses yang dikendalikan oleh komponen. Untuk informasi selengkapnya, lihat Secrets manager.
- Koneksi yang aman antara perangkat dan cloud AWS dengan autentikasi dan otorisasi perangkat. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.

Anda mengonfigurasi dan mengelola perangkat inti Greengrass melalui API AWS IoT Greengrass di mana Anda membuat deployment perangkat lunak berkelanjutan. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat.

Beberapa fitur hanya didukung pada platform tertentu. Untuk informasi selengkapnya, lihat Kompatibilitas fitur Greengrass oleh sistem operasi.

Untuk informasi lebih lanjut tentang platform, persyaratan, dan unduhan yang didukung, lihat Menyiapkan perangkat AWS IoT Greengrass inti.

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

Kompatibilitas fitur Greengrass oleh sistem operasi

AWS IoT Greengrass mendukung perangkat yang menjalankan berbagai sistem operasi. Beberapa fitur hanya didukung pada sistem operasi tertentu. Gunakan tabel berikut untuk mempelajari fitur mana yang tersedia untuk setiap sistem operasi yang didukung. Untuk informasi selengkapnya tentang sistem operasi yang didukung, persyaratan, dan cara menyiapkan perangkat inti Greengrass, lihat. Menyiapkan perangkat AWS IoT Greengrass inti

Perpesanan

Fitur	Linux	Windows
Bertukar pesan MQTT antara dan komponen AWS IoT	 Ya	 Ya
Bertukar pesan penerbita n/berlangganan lokal antar komponen	 Ya	 Ya
Berinteraksi dengan perangkat IoT lokal melalui MQTT	 Ya	 Ya

Fitur	Linux	Windows
Berinteraksi dengan perangkat Modbus-RTU lokal menggunakan komponen Modbus-RTU	 Ya	 Tidak

Keamanan

Fitur	Linux	Windows
Koneksi aman dengan otentikasi dan otorisasi perangkat	 Ya	 Ya
Menyebarkan dan mengakses rahasia yang aman dan terenkripsi dari AWS Secrets Manager	 Ya	 Ya
Gunakan modul keamanan perangkat keras (HSM) untuk menyimpan kunci pribadi dan sertifikat perangkat dengan aman	 Ya	 Tidak
Audit perangkat inti dengan AWS IoT Device Defender	 Ya	 Ya
Gunakan AWS kredensil untuk berinteraksi dengan layanan AWS	 Ya	 Ya

Penginstalan

Fitur	Linux	Windows
Instal AWS IoT Greengrass dengan penyediaan otomatis	 Ya	 Ya
Instal AWS IoT Greengrass dengan penyediaan manual	 Ya	 Ya
Instal AWS IoT Greengrass dengan penyediaan AWS IoT armada	 Ya	 Ya
Instal AWS IoT Greengrass dengan plugin penyediaan khusus	 Ya	 Ya
Jalankan AWS IoT Greengrass dalam wadah Docker menggunakan image Docker bawaan	 Ya	 Tidak

Pemeliharaan dan pembaruan jarak jauh

Fitur	Linux	Windows
Lakukan pembaruan perangkat lunak over-the-air (OTA) yang aman	 Ya	 Ya

Fitur	Linux	Windows
Kelola perangkat inti dengan AWS Systems Manager	 Ya	 Tidak
Connect ke perangkat inti dengan AWS IoT tunneling aman	 Ya	 Tidak

Machine learning

Fitur	Linux	Windows
Lakukan inferensi pembelajaran mesin menggunakan Amazon SageMaker Edge Manager	 Ya	 Ya
Lakukan inferensi pembelajaran mesin menggunakan Amazon Lookout for Vision	 Ya	 Tidak
Lakukan inferensi pembelajaran mesin menggunakan DLR	 Ya	 Ya
Lakukan inferensi pembelajaran mesin menggunakan TensorFlow	 Ya	 Ya

Fitur komponen

Fitur	Linux	Windows
Menyebarkan dan menjalankan fungsi Lambda	 Ya	 Tidak
Jalankan kontainer Docker dalam komponen	 Ya	 Ya
Memproses dan mengekspor aliran data volume tinggi menggunakan pengelola aliran	 Ya	 Ya
Mengelola siklus hidup komponen dengan skrip siklus hidup	 Ya	 Ya
Berinteraksilah dengan bayangan perangkat	 Ya	 Ya
Unggah log ke Amazon CloudWatch Logs	 Ya	 Ya

Fitur	Linux	Windows
Unggah data ke CloudWatch metrik Amazon menggunakan komponen CloudWatch metrik	 Ya	 Ya
Publikasikan pesan ke Amazon Simple Notificat ion Service menggunakan komponen Amazon SNS	 Ya	 Tidak
Publikasikan data ke aliran pengiriman Amazon Data Firehose menggunakan pengelola aliran	 Ya	 Ya
Publikasikan data ke aliran pengiriman Amazon Data Firehose menggunakan komponen Firehose	 Ya	 Tidak
Kumpulkan dan bertindak berdasarkan metrik telemetri sistem waktu nyata	 Ya	 Ya
Konfigurasikan batas sumber daya sistem untuk proses komponen	 Ya	 Tidak
Jeda dan lanjutkan proses komponen	 Ya	 Tidak

Fitur	Linux	Windows
Integrasikan dengan AWS IoT SiteWise menggunakan AWS IoT SiteWise komponen	 Ya	 Ya
Publikasikan aliran video ke Amazon Kinesis Video Streams menggunakan konektor tepi untuk komponen Kinesis Video Streams	 Ya	 Tidak

Pengembangan komponen

Fitur	Linux	Windows
Mengembangkan komponen secara lokal pada perangkat inti	 Ya	 Ya
Berinteraksi dengan perangkat inti menggunakan AWS IoT Greengrass CLI	 Ya	 Ya
Berinteraksi dengan perangkat inti menggunakan konsol debug lokal	 Ya	 Ya
Gunakan AWS IoT Device SDK for Python dalam komponen khusus	 Ya	 Ya

Fitur	Linux	Windows
Gunakan AWS IoT Device SDK untuk C++ dalam komponen khusus	 Ya	 Ya
Gunakan AWS IoT Device SDK untuk Java dalam komponen khusus	 Ya	 Ya

Sertifikasi perangkat

Fitur	Linux	Windows
Gunakan AWS IoT Device Tester for untuk AWS IoT Greengrass V2 memvalidasi perangkat IoT	 Ya	 Ya

Apa yang baru di AWS IoT Greengrass Version 2

AWS IoT Greengrass Version 2 adalah versi utama AWS IoT Greengrass yang memperkenalkan fitur-fitur berikut:

- Komponen yang didukung penerbit - AWS IoT Greengrass sekarang menawarkan komponen yang didukung Penerbit. Komponen ini dikembangkan, ditawarkan, dan dilayani oleh vendor pihak ketiga. Untuk informasi selengkapnya, lihat Komponen yang didukung penerbit.
- Mengoperasikan perangkat Greengrass di VPC — Mengoperasikan perangkat inti Greengrass di VPC sekarang tersedia. Ini memungkinkan Anda untuk melakukan penyebaran di VPC tanpa akses internet publik. Untuk informasi selengkapnya, lihat Mengoperasikan perangkat AWS IoT Greengrass inti di VPC.
- Greengrass Testing Framework (GTF) - GTF untuk sekarang tersedia. AWS IoT Greengrass Version 2 GTF adalah kumpulan blok bangunan untuk mendukung end-to-end otomatisasi. Ini memungkinkan pelanggan AWS IoT Greengrass Version 2 internal untuk menggunakan kerangka pengujian yang sama dengan yang digunakan tim layanan untuk perubahan perangkat lunak yang memenuhi syarat, penerimaan otomatis, dan tujuan jaminan kualitas. Untuk informasi lebih lanjut, lihat Greengrass Testing Framework di Github.
- Bersertifikat PSA — versi AWS IoT Greengrass inti 2.7.0 dan yang lebih baru sekarang bersertifikat Platform Security Architecture (PSA). Untuk informasi lebih lanjut, lihat AWS IoT Greengrass bersertifikat PSA.

AWS IoT Greengrass Catatan rilis memberikan rincian tentang AWS IoT Greengrass rilis—fitur baru, pembaruan dan peningkatan, dan perbaikan umum. AWS IoT Greengrass memiliki jenis rilis berikut:

- Rilis fitur baru untuk AWS IoT Greengrass
- AWS IoT Greengrass Pembaruan perangkat lunak inti

Bagian ini berisi semua catatan AWS IoT Greengrass V2 rilis, terbaru pertama, dan mencakup perubahan fitur utama dan perbaikan bug yang signifikan. Untuk informasi tentang perbaikan kecil tambahan, lihat organisasi aws-greengrass di. GitHub

Catatan rilis

- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.6 pada 24 Mei 2024
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.5 pada 25 April 2024

- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.4 pada 02 April 2024
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.3 pada 27 Maret 2024
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.2 pada 15 Februari 2024
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.1 pada 8 Desember 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.0 pada 7 November 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.11.3 pada 18 Oktober 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.11.2 pada 9 Agustus 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.11.1 pada 21 Juli 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.11.0 pada 28 Juni 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.10.3 pada 21 Juni 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.10.2 pada 5 Juni 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.10.1 pada 11 Mei 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.10.0 pada 9 Mei 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.6 pada 20 April 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.5 pada 30 Maret 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.4 pada 24 Februari 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.3 pada 01 Februari 2023
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.2 pada 22 Desember 2022
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.1 pada 18 November 2022
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.0 pada 15 November 2022
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.8.1 pada 13 Oktober 2022
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.8.0 pada 7 Oktober 2022
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.7.0 pada 28 Juli 2022
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.6.0 pada 27 Juni 2022
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.6 pada 31 Mei 2022
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.5 pada 6 April 2022
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.4 pada 23 Maret 2022
- Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.3 pada 6 Januari 2022

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.6 pada 24 Mei 2024

Rilis ini menyediakan versi 2.12.6 dari komponen inti Greengrass dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 24 Mei 2024

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena itu, versi patch baru dari komponen publik AWS yang disediakan mungkin secara otomatis diterapkan ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup sesuatu, atau Anda memperbarui penerapan yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.

Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak AWS IoT Greengrass Core, lihatPerbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.12.6 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah yang menyebabkan crash saat startup pada prosesor ARMv8 tertentu, termasuk Jetson Nano.
CLI Greengrass	Versi 2.12.6 dari CLI Greengrass tersedia. Perbaikan bug dan peningkatan • Versi diperbarui untuk Greengrass nucleus versi 2.12.6 rilis.
Manajer rahasia	Versi 2.1.8 dari manajer rahasia tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana manajer rahasia tidak menerima sebagian arn.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.5 pada 25 April 2024

Rilis ini menyediakan versi 2.12.5 dari komponen inti Greengrass dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 25 April 2024

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena itu, versi patch baru dari komponen publik AWS yang disediakan mungkin secara otomatis diterapkan ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup sesuatu, atau Anda memperbarui penerapan yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak AWS IoT Greengrass Core, lihatPerbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.12.5 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana rollback penerapan terkadang macet saat memutar kembali komponen yang sebelumnya rusak dengan dependensi keras. • Memperbaiki masalah di mana nukleus tidak mempublikasikan pembaruan status setelah penyediaan armada. • Menambahkan percobaan ulang untuk `GetDeploymentConfiguration` API setelah mendapatkan 404 kesalahan.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.4 pada 02 April 2024

Rilis ini menyediakan versi 2.12.4 komponen inti Greengrass dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 02 April 2024

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena itu, versi patch baru dari komponen publik AWS yang disediakan mungkin secara otomatis diterapkan ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup sesuatu, atau Anda memperbarui penerapan yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak AWS IoT Greengrass Core, lihatPerbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.12.4 dari inti Greengrass tersedia.

Komponen	Detail
	Perbaikan bug dan peningkatan • Memperbaiki masalah saat nukleus memasuki kondisi kebuntuan selama startup di beberapa perangkat Linux.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.3 pada 27 Maret 2024

Rilis ini menyediakan versi 2.12.3 dari komponen inti Greengrass dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 27 Maret 2024

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena itu, versi patch baru dari komponen publik AWS yang disediakan mungkin secara otomatis diterapkan ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup sesuatu, atau Anda memperbarui penerapan yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak AWS IoT Greengrass Core, lihatPerbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.12.3 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana nukleus tidak melaporkan status komponen yang benar setelah nukleus diluncurkan kembali dan selama pemulihan komponen. • Perbaikan bug umum dan perbaikan.
Manajer bayangan	Versi 2.3.7 dari komponen shadow manager tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah saat pengelola bayangan mencatat `NullPointerException` kesalahan secara berkala selama sinkronisasi pengelola bayangan.
Penyediaan armada	Versi 1.2.1 dari plugin penyediaan AWS IoT armada tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah saat plugin penyediaan armada sedang offline selama startup inti Greengrass. Plugin penyediaan armada sekarang tanpa batas waktu mencoba ulang panggilan koneksi MQTT.
Detektor IP	Versi 2.1.9 dari komponen spooler disk tersedia. Perbaikan bug dan peningkatan Menyesuaikan langkah yang diperoleh IP untuk hanya mengirim log pada tingkat log debug.
Moquette MQTT 3.1.1 komponen broker	Versi 2.3.6 dari komponen broker Moquette MQTT 3.1.1 tersedia. Perbaikan bug dan peningkatan Perbaikan bug umum dan perbaikan.
Manajer Lambda	Versi 2.3.3 dari komponen manajer Lambda tersedia.

Komponen	Detail
	Perbaikan bug dan peningkatan Perbaikan bug umum dan perbaikan.
Konsol debug lokal	Versi 2.4.2 dari komponen konsol debug lokal tersedia. Perbaikan bug dan peningkatan Perbaikan bug umum dan perbaikan.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.2 pada 15 Februari 2024

Rilis ini menyediakan versi 2.12.2 dari komponen inti Greengrass dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 15 Februari 2024

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena itu, versi patch baru dari komponen publik AWS yang disediakan mungkin secara otomatis diterapkan ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup sesuatu, atau Anda memperbarui penerapan yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung

saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak AWS IoT Greengrass Core, lihatPerbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.12.2 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana log lama tidak dibersihkan dengan benar. • Perbaikan bug umum dan perbaikan.
Manajer bayangan	Versi 2.3.6 dari komponen shadow manager tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah di mana properti bayangan yang dihapus melalui AWS Cloud pembaruan saat perangkat offline terus ada di bayangan lokal setelah mendapatkan kembali konektivitas.
Peluncur Lambda	Versi 2.0.13 dari komponen peluncur lambda tersedia. Perbaikan bug dan peningkatan Perbaikan bug umum dan perbaikan.
Spooler disk	Versi 1.0.3 dari komponen spooler disk tersedia. Perbaikan bug dan peningkatan Meningkatkan kinerja dengan menggunakan kembali koneksi database.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.1 pada 8 Desember 2023

Rilis ini menyediakan versi 2.12.1 dari komponen inti Greengrass dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 8 Desember 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.12.1 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana nukleus dapat menduplikasi langganan MQTT ke topik penerapan yang mengarah ke pencatatan tambahan dan penerbitan MQTT.
Autentikasi perangkat klien	Versi 2.4.5 komponen autentikasi perangkat klien tersedia.

Komponen	Detail
	Fitur baru Menambahkan dukungan untuk awalan wildcard untuk memilih nama benda dengan parameter. `selectionRule` Perbaikan bug dan peningkatan Memperbaiki masalah saat sertifikat tidak diperbarui dengan informasi konektivitas baru dalam kasus tertentu.
Spooler disk	Versi 1.0.2 dari komponen spooler disk tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah di mana bidang format pesan MQTT tidak bertahan dalam kasus tertentu.
Jembatan MQTT	Versi 2.3.1 dari komponen spooler disk tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah saat klien MQTT lokal masuk ke loop pemutusan.
Manajer aliran	Versi 2.1.12 dari komponen manajer aliran tersedia. Perbaikan bug dan peningkatan Memperbarui urutan kredensil yang digunakan sehingga kredenal Greengrass lebih disukai untuk permintaan layanan. AWS

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.12.0 pada 7 November 2023

Rilis ini menyediakan versi 2.12.0 dari komponen inti Greengrass dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 7 November 2023

Sorotan rilis

- Bootstrap on rollback — AWS IoT Greengrass sekarang menyediakan parameter konfigurasi inti Greengrass yang disebut. `BootstrapOnRollback` Fitur ini memungkinkan Anda untuk menjalankan langkah-langkah siklus hidup bootstrap sebagai bagian dari penerapan rollback.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.12.0 dari inti Greengrass tersedia. Fitur baru • Memungkinkan Anda menjalankan langkah-langkah siklus hidup bootstrap sebagai bagian dari penerapan rollback.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.11.3 pada 18 Oktober 2023

Rilis ini menyediakan versi 2.11.3 dari komponen inti Greengrass.

Tanggal rilis: 18 Oktober 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.11.3 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah di nukleus di mana ia mungkin memulai komponen dengan tidak benar ketika dependensinya gagal.

Komponen	Detail
	Fitur baru • Menambahkan tipe endpoint s3 yang dapat dikonfigurasi.
Manajer Lambda	Versi 2.3.1 dari komponen manajer Lambda tersedia. Perbaikan bug dan peningkatan • Menyesuaikan tingkat log untuk kesalahan tertentu.
Konsol deubg lokal	Versi 2.4.0 dari komponen manajer Lambda tersedia. Fitur baru • Menambahkan konsol debugging manajer aliran.
Manajer log	Versi 2.3.6 dari komponen pengelola log tersedia. Perbaikan bug dan peningkatan • Menyesuaikan tingkat log untuk kesalahan tertentu.
Manajer bayangan	Versi 2.3.4 dari komponen Shadow manager tersedia. Perbaikan bug dan peningkatan • Menambahkan dukungan untuk dokumen status bayangan nol dan kosong.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.11.2 pada 9 Agustus 2023

Rilis ini menyediakan versi 2.11.2 dari komponen inti Greengrass.

Tanggal rilis: 9 Agustus 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.11.2 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah di klien nukleus MQTT 5 yang mungkin muncul offline saat sejumlah besar (> 50) langganan sedang digunakan. • Menambahkan percobaan lagi untuk kegagalan TCP docker dial.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.11.1 pada 21 Juli 2023

Rilis ini menyediakan versi 2.11.1 dari komponen inti Greengrass.

Tanggal rilis: 21 Juli 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.11.1 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana nukleus tidak dimulai jika tugas bootstrap gagal dan file metadata penerapan rusak. • Memperbaiki masalah di mana komponen Lambda sesuai permintaan tidak dilaporkan dalam pembaruan status penerapan. • Menambahkan dukungan untuk ID kebijakan otorisasi duplikat.
Manajer Lambda	Versi 2.2.11 dari manajer Lambda tersedia.

Komponen	Detail
	Perbaikan bug dan peningkatan • Memperbaiki masalah saat LegacySubscriptionRouter konfigurasi tidak diperbarui saat konfigurasi Lambda berubah.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.11.0 pada 28 Juni 2023

Rilis ini menyediakan versi 2.11.0 dari komponen inti Greengrass.

Tanggal rilis: 28 Juni 2023

Sorotan rilis

- Persistent disk spooler — AWS IoT Greengrass sekarang menyediakan implementasi spooler persisten untuk pesan yang di-spooled dari perangkat inti Greengrass ke. AWS IoT Core Komponen ini akan menyimpan pesan keluar ini pada disk. Untuk informasi selengkapnya, lihat Spooler disk.
- Peningkatan penerapan lokal — Anda sekarang dapat membatalkan penerapan lokal, mengatur kebijakan penanganan kegagalan penerapan, dan mendapatkan status penerapan terperinci.
- Peningkatan kecepatan logging - Kecepatan unggah log untuk komponen pengelola log telah ditingkatkan.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen

publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.11.0 dari inti Greengrass tersedia. Fitur baru • Memungkinkan Anda membatalkan penerapan lokal. • Memungkinkan Anda mengonfigurasi kebijakan penanganan kegagalan untuk penerapan lokal. • Menambahkan dukungan untuk plugin spooler disk.
CLI Greengrass	Versi 2.11.0 dari CLI Greengrass tersedia. Fitur baru • Memungkinkan Anda membatalkan penerapan lokal. • Memungkinkan Anda mengonfigurasi kebijakan penanganan kegagalan untuk penerapan lokal. • Meningkatkan pelaporan status penyebaran terperinci.
Spooler disk	Versi 1.0.0 dari komponen spooler disk tersedia. • Komponen spooler disk menyediakan penyimpanan persisten pesan yang dikirim dari perangkat inti Greengrass ke. AWS IoT Core
Manajer log	Versi 2.3.5 dari komponen pengelola log tersedia.

Komponen	Detail
	Perbaikan Meningkatkan kecepatan unggah log.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.10.3 pada 21 Juni 2023

Rilis ini menyediakan versi 2.10.3 dari komponen inti Greengrass.

Tanggal rilis: 21 Juni 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.10.3 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah saat Greengrass tidak berlangganan notifikasi penerapan saat menggunakan penyedia PKCS #11.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.10.2 pada 5 Juni 2023

Rilis ini menyediakan versi 2.10.2 dari komponen inti Greengrass.

Tanggal rilis: 5 Juni 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

Important

Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan

untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.10.2 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memungkinkan penguraian siklus hidup komponen yang tidak peka huruf besar/kecil. • Memperbaiki masalah di mana variabel PATH lingkungan tidak dibuat ulang dengan benar. • Memperbaiki pengkodean URI proxy untuk komponen termasuk manajer aliran untuk nama pengguna dengan karakter khusus.
Autentikasi perangkat klien	Versi 2.4.2 komponen autentikasi perangkat klien tersedia. Fitur baru Menambahkan opsi `startupTimeoutSeconds` konfigurasi baru.
Manajer Lambda	Versi 2.2.9 dari komponen manajer Lambda tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah di mana nomor port rusak karena jam miring.
Manajer log	Versi 2.3.4 dari komponen pengelola log tersedia. Perbaikan bug dan peningkatan • Menambahkan dukungan untuk mengatur `periodicUploadIntervalSec` parameter ke nilai fraksional. Minimal adalah 1 mikrodetik. • Memperbaiki masalah di mana pengelola log tidak menghormati CloudWatch `putLogEvents` batas.
MQTT 3.1 broker (Moquette)	Versi 2.3.3 dari komponen broker MQTT 3.1 (Moquette) tersedia.

Komponen	Detail
	Fitur baru Menambahkan opsi `startupTimeoutSeconds` konfigurasi baru.
Jembatan MQTT	Versi 2.2.6 dari komponen jembatan MQTT tersedia. Fitur baru Menambahkan opsi `startupTimeoutSeconds` konfigurasi baru.
Manajer aliran	Versi 2.1.7 dari komponen manajer aliran tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah di mana manajer aliran gagal membaca konfigurasi proxy dengan benar.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.10.1 pada 11 Mei 2023

Rilis ini menyediakan versi 2.10.1 dari komponen inti Greengrass.

Tanggal rilis: 11 Mei 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda

jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.10.1 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah yang dapat menyebabkan crash saat startup pada prosesor ARMv8 tertentu, termasuk Jetson Nano. • Greengrass tidak lagi menutup standar komponen di, ini mengembalikan perilaku ke perilaku pra-2.10.0
Manajer aliran	Versi 2.1.6 dari manajer aliran baru tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah yang dapat menyebabkan crash saat startup pada prosesor ARMv8 tertentu, termasuk Jetson Nano.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.10.0 pada 9 Mei 2023

Rilis ini menyediakan versi 2.10.0 dari komponen inti Greengrass dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 9 Mei 2023

Sorotan rilis

- Dukungan MQTT5 - AWS IoT Greengrass sekarang mendukung pengiriman dan penerimaan pesan dari AWS IoT Core menggunakan MQTT5. Untuk informasi selengkapnya, lihat Menerbitkan AWS IoT Core pesan MQTT.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.10.0 dari inti Greengrass tersedia.

Komponen	Detail
	Fitur baru • Menambahkan `interpolateComponentConfiguration` dukungan untuk ekspresi reguler kosong. Greengrass sekarang diinterpolasi dari objek konfigurasi root. • Menambahkan dukungan untuk MQTT5. • Menambahkan mekanisme untuk memuat komponen plugin dengan cepat tanpa memindai. • Mengaktifkan Greengrass untuk menghemat ruang disk dengan menghapus gambar Docker yang tidak digunakan. Perbaikan bug dan peningkatan • Memperbaiki masalah saat rollback meninggalkan nilai konfigurasi tertentu dari penerapan. • Memperbaiki masalah di mana inti Greengrass memvalidasi urutan domain dalam non-kredensi kustom AWS dan titik akhir data. AWS • Memperbarui resolusi ketergantungan multi-grup untuk menyelesa ikan kembali semua dependensi grup melalui AWS Cloud negosiasi, alih-alih mengunci ke versi aktif. Pembaruan ini juga menghapus kode `INSTALLED_COMPONENT_NOT_FOUND` kesalahan penerapan. • Memperbarui inti Greengrass untuk melewatkan pengunduhan gambar Docker saat sudah ada secara lokal. • Memperbarui inti Greengrass untuk memulai ulang langkah pemasanga n komponen sebelum batas waktu berakhir. • Peningkatan dan perbaikan kecil tambahan.
Manajer bayangan	Versi 2.3.2 dari manajer bayangan baru tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah saat pengelola bayangan memasuki `BROKEN` status saat database bayangan lokal rusak.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.6 pada 20 April 2023

Rilis ini menyediakan versi 2.9.6 dari komponen inti Greengrass.

Tanggal rilis: 20 April 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.9.6 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah saat penerapan Greengrass gagal dengan kesalahan LAUNCH_DIRECTORY_CORRUPTED dan reboot perangkat

Komponen	Detail
	berikutnya gagal memulai Greengrass. Kesalahan ini dapat terjadi saat Anda memindahkan perangkat Greengrass di antara beberapa grup hal dengan penerapan yang memerlukan Greengrass untuk memulai ulang.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.5 pada 30 Maret 2023

Rilis ini menyediakan versi 2.9.5 dari komponen inti Greengrass.

Tanggal rilis: 30 Maret 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.9.5 dari inti Greengrass tersedia. Fitur baru • Menambahkan dukungan untuk verifikasi tanda tangan perangkat lunak inti Greengrass. Perbaikan bug dan peningkatan • Memperbaiki masalah saat penerapan gagal saat wilayah metadata resep lokal tidak cocok dengan wilayah peluncuran inti Greengrass. Inti Greengrass sekarang bernegosiasi ulang dengan awan ketika ini terjadi. • Memperbaiki masalah saat spooler pesan MQTT terisi dan tidak pernah menghapus pesan. • Peningkatan dan perbaikan kecil tambahan.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.4 pada 24 Februari 2023

Rilis ini menyediakan versi 2.9.4 dari komponen inti Greengrass.

Tanggal rilis: 24 Februari 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda

jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.9.4 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memeriksa pesan nol sebelum menjatuhkan pesan QOS 0. • Mempotong nilai detail status pekerjaan jika melebihi batas karakter 1024. • Memperbarui skrip bootstrap untuk Windows untuk membaca jalur root Greengrass dengan benar jika jalur itu menyertakan spasi. • Pembaruan berlangganan AWS IoT Core sehingga menghapus pesan klien jika respons langganan tidak dikirim. • Memastikan bahwa nukleus memuat konfigurasinya dari file cadangan saat file konfigurasi utama rusak atau hilang.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.3 pada 01 Februari 2023

Rilis ini menyediakan versi 2.9.3 dari komponen inti Greengrass.

Tanggal rilis: 01 Februari 2023

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan yang diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.9.3 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memastikan ID klien MQTT tidak diduplikasi. • Menambahkan pembacaan dan penulisan file yang lebih kuat untuk menghindari dan memulihkan dari korupsi. • Mencoba ulang gambar docker menarik kesalahan terkait jaringan tertentu. • Menambahkan `noProxyAddresses` opsi untuk koneksi MQTT.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.2 pada 22 Desember 2022

Rilis ini menyediakan versi 2.9.2 dari komponen inti Greengrass.

Tanggal rilis: 22 Desember 2022

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.9.2 dari inti Greengrass tersedia.

Komponen	Detail
	Perbaikan bug dan peningkatan • Memperbaiki masalah saat konfigurasi `interpolateComponentConfiguration` tidak berlaku untuk penerapan yang sedang berlangsung. • Menggunakan OSHI untuk membuat daftar semua proses anak.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.1 pada 18 November 2022

Rilis ini menyediakan versi 2.9.1 dari komponen inti Greengrass dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 18 November 2022

Sorotan rilis

- Manajer log — Manajer log sekarang memproses dan langsung mengunggah file log aktif alih-alih menunggu file baru diputar. Peningkatan ini secara signifikan mengurangi penundaan log. Untuk informasi selengkapnya, lihat Manajer log

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment

yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.9.1 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Menambahkan perbaikan di mana Greengrass dimulai ulang jika penerapan menghapus komponen plugin.
Manajer log	Versi 2.3.0 dari pengelola log baru tersedia. **Note** Kami menyarankan Anda meningkatkan ke Greengrass nucleus 2.9.1 saat Anda meningkatkan ke pengelola log 2.3.0. Fitur baru • Mengurangi penundaan log dengan memproses dan langsung mengunggah file log aktif alih-alih menunggu file baru diputar. Perbaikan bug dan peningkatan • Meningkatkan dukungan rotasi log saat memutar file dengan nama unik. • Peningkatan dan perbaikan kecil tambahan.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.9.0 pada 15 November 2022

Rilis ini menyediakan versi 2.9.0 dari komponen inti Greengrass dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 15 November 2022

Sorotan rilis

- Otentikasi offline - AWS IoT Greengrass sekarang mendukung otentikasi offline. Anda dapat mengonfigurasi perangkat AWS IoT Greengrass inti Anda sehingga perangkat klien dapat terhubung ke perangkat inti, bahkan ketika perangkat inti tidak terhubung ke cloud. Untuk informasi selengkapnya, lihat Autentikasi offline.
- Subdeployments — Anda sekarang dapat membuat subdeployments. Anda dapat menggunakan subdeployment untuk menyelesaikan penerapan yang gagal. Setiap subdeployment dapat menguji konfigurasi yang berbeda dari penerapan yang gagal pada subset perangkat yang lebih kecil. Untuk informasi selengkapnya, lihat Membuat subdeployments.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung

saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.9.0 dari inti Greengrass tersedia. Fitur baru • Menambahkan kemampuan untuk membuat subdeployment yang mencoba lagi penerapan dengan subset perangkat yang lebih kecil. Fitur ini menciptakan cara yang lebih efisien untuk menguji dan menyelesaikan penerapan yang gagal. Perbaikan bug dan peningkatan • Meningkatkan dukungan untuk sistem yang tidak memiliki`useradd`,`groupadd`, dan`usermod`. • Peningkatan dan perbaikan kecil tambahan.
Autentikasi perangkat klien	Versi 2.3.0 komponen autentikasi perangkat klien tersedia. Fitur baru • Menambahkan dukungan untuk otentikasi offline perangkat klien. Dengan fitur ini, perangkat klien dapat terus terhubung ke perangkat inti ketika perangkat inti tidak terhubung ke Internet. • Menambahkan dukungan untuk otoritas sertifikat yang disediakan pelanggan (CA). Perangkat inti Anda menggunakan CA yang disediaka n pelanggan sebagai sertifikat root untuk menghasilkan sertifikat broker MQTT.
Pialang MQTT 5 (EMQX)	Versi 1.2.0 Pialang MQTT 5 (EMQX) komponen tersedia. Fitur baru Menambahkan dukungan untuk rantai sertifikat.

Komponen	Detail
Pialang MQTT Moquette	Versi 2.3.0 dari komponen broker Moquette MQTT baru tersedia. Fitur baru Menambahkan dukungan untuk rantai sertifikat.
Manajer rahasia	Versi 2.1.4 dari manajer rahasia baru tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah saat rahasia yang di-cache dihapus saat manajer rahasia dikerahkan dan inti Greengrass dimulai ulang.
Manajer aliran	Versi 2.1.2 dari manajer aliran baru tersedia. Perbaikan bug dan peningkatan Memperbaiki masalah pada OS Windows yang menggunakan bahasa non-Inggris.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.8.1 pada 13 Oktober 2022

Rilis ini menyediakan versi 2.8.1 dari komponen inti Greengrass.

Tanggal rilis: 13 Oktober 2022

> **Note**
>
> Jika Anda menggunakan Greengrass nucleus versi 2.8.0, kami sangat menyarankan Anda meningkatkan ke Greengrass nucleus versi 2.8.1.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.8.1 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah saat kode kesalahan penerapan tidak dihasilkan dengan benar dari kesalahan Greengrass API. • Memperbaiki masalah saat pembaruan status armada mengirimkan informasi yang tidak akurat saat komponen mencapai `ERRORED` status selama penerapan. • Memperbaiki masalah di mana penerapan tidak dapat diselesaikan ketika Greengrass memiliki lebih dari 50 langganan yang ada.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.8.0 pada 7 Oktober 2022

Rilis ini menyediakan versi 2.8.0 dari komponen inti Greengrass dan versi 1.1.0 dari komponen broker MQTT 5 (EMQX).

Tanggal rilis: 7 Oktober 2022

Sorotan rilis

- Kode kesalahan penerapan — Inti Greengrass sekarang melaporkan respons status kesehatan penerapan yang menyertakan kode kesalahan terperinci saat penerapan komponen tidak dapat diselesaikan. Untuk informasi selengkapnya, lihat Kode kesalahan penyebaran terperinci.
- Status kesalahan komponen — Inti Greengrass sekarang melaporkan respons status kesehatan komponen yang mencakup status kesalahan terperinci saat komponen memasuki status atau. `BROKEN` `ERRORED` Untuk informasi selengkapnya, lihat Kode status komponen rinci.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan

untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.8.0 dari inti Greengrass tersedia. Fitur baru • Memperbarui inti Greengrass untuk melaporkan respons status kesehatan penerapan yang mencakup kode kesalahan terperinci ketika ada masalah dalam menerapkan komponen ke perangkat inti. Untuk informasi selengkapnya, lihat Kode kesalahan penyebaran terperinci. • Memperbarui inti Greengrass untuk melaporkan respons status kesehatan komponen yang mencakup kode kesalahan terperinci saat komponen memasuki status atau. BROKEN ERRORED Untuk informasi selengkapnya, lihat Kode status komponen rinci. • Memperluas bidang pesan status untuk meningkatkan informasi ketersediaan cloud untuk perangkat. • Meningkatkan kekokohan layanan status armada. Perbaikan bug dan peningkatan • Memungkinkan komponen yang rusak untuk menginstal ulang ketika konfigurasinya berubah. • Memperbaiki masalah saat nukleus dimulai ulang selama penerapan bootstrap menyebabkan penerapan gagal. • Memperbaiki masalah di Windows di mana instalasi gagal ketika jalur root berisi spasi. • Memperbaiki masalah saat komponen dimatikan selama penerapan menggunakan skrip shutdown versi baru. • Berbagai peningkatan shutdown. • Peningkatan dan perbaikan kecil tambahan.
Pialang MQTT 5 (EMQX)	Versi 1.1.0 Pialang MQTT 5 (EMQX) komponen tersedia.

Komponen	Detail
	Fitur baru • Menambahkan dukungan untuk konfigurasi EMQX termasuk opsi broker dan plug-in. Perbaikan bug dan peningkatan • Pembaruan EMQX ke versi 4.4.9.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.7.0 pada 28 Juli 2022

Rilis ini menyediakan versi 2.7.0 dari komponen inti Greengrass, versi 2.1.0 dari komponen manajer aliran, dan versi 2.2.5 dari komponen manajer Lambda.

Tanggal rilis: 28 Juli 2022

Sorotan rilis

- Metrik telemetri manajer streaming — Manajer streaming sekarang secara otomatis mengirimkan metrik telemetri ke EventBridge Amazon, sehingga Anda dapat membuat aplikasi cloud yang memantau dan menganalisis volume data yang diunggah perangkat inti Anda. Untuk informasi selengkapnya, lihat Kumpulkan data telemetri kondisi sistem dari perangkat inti AWS IoT Greengrass.
- Custom Certificate Authority (CA) — Sertifikat klien yang ditandatangani oleh CA sertifikat kustom, di mana CA tidak terdaftarAWS IoT, sekarang didukung. Untuk informasi selengkapnya, lihat Menggunakan sertifikat perangkat yang ditandatangani oleh CA pribadi.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.7.0 dari inti Greengrass tersedia. Fitur baru • Memperbarui inti Greengrass untuk mengirim pembaruan status ke cloud saat perangkat AWS IoT Greengrass inti menerapkan penerapan lokal. • Menambahkan dukungan untuk sertifikat klien yang ditandatangani oleh otoritas sertifikat khusus (CA), di mana CA tidak terdaftarAWS IoT. Untuk menggunakan fitur ini, Anda dapat mengatur opsi `greengrassDataPlaneEndpoint` konfigurasi baru `keiotdata`. Untuk informasi selengkapnya, lihat Menggunakan sertifikat perangkat yang ditandatangani oleh CA pribadi. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana inti Greengrass memutar kembali penerapan dalam skenario tertentu saat nukleus dihentikan atau dimulai ulang. Nukleus sekarang melanjutkan penyebaran setelah nukleus dimulai kembali.

Komponen	Detail
	• Memperbarui penginstal Greengrass untuk menghormati `--start` argumen saat Anda menentukan untuk mengatur perangkat lunak sebagai layanan sistem. • Memperbarui perilaku SubscribeToComponentUpdates untuk menyetel ID penerapan dalam peristiwa di mana nukleus memperbarui komponen. • Peningkatan dan perbaikan kecil tambahan.
Manajer aliran	Versi 2.1.0 dari komponen manajer aliran tersedia. Fitur baru • Memperbarui komponen ini untuk secara otomatis mengirim metrik telemetri ke Amazon. EventBridge Untuk informasi selengkapnya, lihat Kumpulkan data telemetri kondisi sistem dari perangkat inti AWS IoT Greengrass. Fitur ini membutuhkan v2.7.0 atau yang lebih baru dari komponen inti Greengrass. • Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
Manajer Lambda	Versi 2.2.5 dari komponen manajer Lambda tersedia. Fitur baru • Menambahkan dukungan untuk wildcard topik MQTT di sumber acara tempat Anda berlangganan pesan penerbitan/berlangganan lokal. Fitur ini membutuhkan v2.6.0 atau yang lebih baru dari komponen inti Greengrass. • Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.6.0 pada 27 Juni 2022

Rilis ini menyediakan versi 2.6.0 dari komponen inti Greengrass, komponen baru yang disediakan, dan pembaruan ke komponen AWS yang disediakan. AWS

Tanggal rilis: 27 Juni 2022

Sorotan rilis

- Wildcard dalam topik penerbitan/berlangganan lokal - Anda sekarang dapat menggunakan wildcard MQTT saat Anda berlangganan topik penerbitan/berlangganan lokal. Lihat informasi yang lebih lengkap di Pesan lokal publikasi/berlangganan dan SubscribeToTopic.
- Dukungan bayangan perangkat klien - Anda sekarang dapat berinteraksi dengan bayangan perangkat klien dalam komponen khusus dan menyinkronkan bayangan perangkat klien denganAWS IoT Core. Untuk informasi selengkapnya, lihat Berinteraksi dengan dan menyinkronkan bayangan perangkat klien.
- Dukungan MQTT 5 lokal untuk perangkat klien - Anda sekarang dapat menggunakan broker EMQX MQTT 5 untuk menggunakan fitur MQTT 5 dalam komunikasi antara perangkat klien dan perangkat inti. Lihat informasi yang lebih lengkap di Pialang MQTT 5 (EMQX) dan Hubungkan perangkat klien ke perangkat inti.
- Variabel resep dalam konfigurasi komponen — Anda sekarang dapat menggunakan variabel resep tertentu dalam konfigurasi komponen. Anda dapat menggunakan variabel resep ini saat menentukan konfigurasi default komponen dalam resep atau saat Anda mengonfigurasi komponen dalam penerapan. Lihat informasi yang lebih lengkap di Variabel resep dan Gunakan variabel resep dalam menggabungkan pembaruan.
- Wildcard dalam kebijakan otorisasi IPC — Anda sekarang dapat menggunakan * wildcard untuk mencocokkan kombinasi karakter apa pun dalam kebijakan otorisasi komunikasi antar proses (IPC). Wildcard ini memungkinkan Anda untuk mengizinkan akses ke beberapa sumber daya dalam satu kebijakan otorisasi. Untuk informasi selengkapnya, lihat Wildcard dalam kebijakan otorisasi.
- Operasi IPC yang mengelola penerapan dan komponen lokal — Anda sekarang dapat mengembangkan komponen kustom yang mengelola penerapan lokal dan melihat detail komponen. Untuk informasi selengkapnya, lihat IPC: Mengelola penerapan dan komponen lokal.

- Operasi IPC yang mengautentikasi dan mengotorisasi perangkat klien — Anda sekarang dapat menggunakan operasi ini untuk membuat komponen broker lokal kustom. Untuk informasi selengkapnya, lihat IPC: Mengautentikasi dan mengotorisasi perangkat klien.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.6.0 dari inti Greengrass tersedia. Fitur baru • Menambahkan dukungan untuk wildcard MQTT saat Anda berlangga nan topik penerbitan/berlangganan lokal. Lihat informasi yang lebih lengkap di Pesan lokal publikasi/berlangganan dan SubscribeToTopic.

Komponen	Detail
	• Menambahkan dukungan untuk variabel resep dalam konfigura si komponen, selain variabel *component_dependen cy_name* :configuration: *json_pointer* resep. Anda dapat menggunakan variabel resep ini saat menentukan komponen DefaultConfiguration dalam resep atau saat Anda mengonfig urasi komponen dalam penerapan. Untuk mengaktifkan fitur ini, atur opsi interpolateComponentConfigurationkonfigurasi ketrue. Lihat informasi yang lebih lengkap di Variabel resep dan Gunakan variabel resep dalam menggabungkan pembaruan. • Menambahkan dukungan penuh untuk kebijakan * otorisasi wildcard in interprocess communication (IPC). Anda sekarang dapat menentuka n * karakter dalam string sumber daya untuk mencocokkan kombinasi karakter apa pun. Untuk informasi selengkapnya, lihat Wildcard dalam kebijakan otorisasi. • Menambahkan dukungan untuk komponen khusus untuk memanggil operasi IPC yang digunakan CLI Greengrass. Anda dapat menggunak an operasi IPC ini untuk mengelola penerapan lokal, melihat detail komponen, dan membuat kata sandi yang dapat Anda gunakan untuk masuk ke konsol debug lokal. Untuk informasi selengkapnya, lihat IPC: Mengelola penerapan dan komponen lokal. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana komponen dependen tidak akan bereaksi ketika dependensi kerasnya memulai ulang atau mengubah status dalam skenario tertentu. • Memperbaiki pesan kesalahan yang dilaporkan perangkat inti ke layanan AWS IoT Greengrass cloud saat penerapan gagal. • Memperbaiki masalah di mana inti Greengrass menerapkan penerapan sesuatu dua kali dalam skenario tertentu saat nukleus dimulai ulang. • Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.

Komponen	Detail
Pialang MQTT 5 (EMQX)	Versi 1.0.0 dari komponen broker EMQX MQTT 5 baru tersedia. Fitur baru • Menambahkan dukungan untuk broker EMQX MQTT 5 lokal. Perangkat klien dapat terhubung ke broker MQTT ini untuk berkomunikasi dengan perangkat inti menggunakan fitur MQTT 5.
Manajer bayangan	Versi 2.2.0 dari komponen shadow manager tersedia. Fitur baru • Menambahkan dukungan untuk layanan bayangan lokal melalui antarmuka penerbitan/berlangganan lokal. Anda sekarang dapat berkomunikasi dengan broker pesan penerbitan/berlangganan lokal pada topik bayangan MQTT untuk mendapatkan, memperbarui, dan menghapus bayangan pada perangkat inti. Fitur ini memungkinkan Anda untuk menghubungkan perangkat klien ke layanan bayangan lokal dengan menggunakan jembatan MQTT untuk menyampaikan pesan tentang topik bayangan antara perangkat klien dan antarmuka penerbita n/berlangganan lokal. Fitur ini membutuhkan v2.6.0 atau yang lebih baru dari komponen inti Greengrass. Untuk menghubungkan perangkat klien ke layanan bayangan lokal, Anda juga harus menggunakan v2.2.0 atau yang lebih baru dari komponen jembatan MQTT. • Menambahkan `direction` opsi yang dapat Anda konfigurasikan untuk menyesuaikan arah untuk menyinkronkan bayangan antara layanan bayangan lokal danAWS Cloud. Anda dapat mengonfigurasi opsi ini untuk mengurangi bandwidth dan koneksi ke fileAWS Cloud.

Komponen	Detail
Autentikasi perangkat klien	Versi 2.2.0 komponen autentikasi perangkat klien tersedia. Fitur baru • Menambahkan dukungan untuk komponen kustom untuk memanggil operasi komunikasi antarproses (IPC) untuk mengautentikasi dan mengotorisasi perangkat klien. Anda dapat menggunakan operasi ini dalam komponen broker MQTT khusus, misalnya. Untuk informasi selengkapnya, lihat IPC: Mengautentikasi dan mengotorisasi perangkat klien. • Menambahkan`maxActiveAuthTokens` ,`cloudQueueSize` , dan `threadPoolSize` opsi yang dapat Anda konfigurasikan untuk menyetel kinerja komponen ini.
Jembatan MQTT	Versi 2.2.0 dari komponen jembatan MQTT tersedia. Fitur baru • Menambahkan dukungan untuk wildcard topik MQTT (#dan+) saat Anda menentukan publish/subscribe lokal sebagai broker pesan sumber. Fitur ini membutuhkan v2.6.0 atau yang lebih baru dari komponen inti Greengrass. • Menambahkan `targetTopicPrefix` opsi, yang dapat Anda tentukan untuk mengonfigurasi jembatan MQTT untuk menambahkan awalan ke topik target saat menyampaikan pesan.

Komponen	Detail
CLI Greengrass	Versi 2.6.0 dari CLI Greengrass tersedia. Fitur baru • Menambahkan dukungan untuk komponen khusus untuk memanggil operasi komunikasi antarproses (IPC) yang digunakan CLI Greengrass. Anda dapat menggunakan operasi IPC ini untuk mengelola penerapan lokal, melihat detail komponen, dan membuat kata sandi yang dapat Anda gunakan untuk masuk ke konsol debug lokal. Untuk informasi selengkapnya, lihat IPC: Mengelola penerapan dan komponen lokal. Perbaikan bug dan peningkatan • Peningkatan dan perbaikan kecil tambahan.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.6 pada 31 Mei 2022

Rilis ini menyediakan versi 2.5.6 komponen inti Greengrass dan versi 2.2.4 dari komponen pengelola log.

Tanggal rilis: 31 Mei 2022

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment

yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.5.6 dari inti Greengrass tersedia. Fitur baru • Menambahkan dukungan untuk modul keamanan perangkat keras yang menggunakan kunci ECC. Anda dapat menggunakan modul keamanan perangkat keras (HSM) untuk menyimpan kunci pribadi dan sertifikat perangkat dengan aman. Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana penerapan tidak pernah selesai saat Anda menerapkan komponen dengan skrip penginstalan yang rusak dalam skenario tertentu. • Meningkatkan kinerja selama startup. • Peningkatan dan perbaikan kecil tambahan.
Manajer log	Versi 2.2.4 dari komponen pengelola log tersedia. Perbaikan bug dan peningkatan • Meningkatkan stabilitas saat menangani konfigurasi yang tidak valid. • Peningkatan dan perbaikan kecil tambahan.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.5 pada 6 April 2022

Rilis ini menyediakan versi 2.5.5 dari komponen inti Greengrass.

Tanggal rilis: 6 April 2022

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.5.5 dari inti Greengrass tersedia.

Komponen	Detail
	Fitur baru • Menambahkan variabel `GG_ROOT_CA_PATH` lingkungan untuk komponen, sehingga Anda dapat mengakses sertifikat root certificate authority (CA) di komponen kustom. Perbaikan bug dan peningkatan • Menambahkan dukungan untuk perangkat Windows yang menggunakan bahasa tampilan selain bahasa Inggris. • Memperbarui cara inti Greengrass mem-parsing argumen installer Boolean, sehingga Anda dapat menentukan argumen Boolean tanpa nilai Boolean untuk menentukan nilai. `true` Misalnya, Anda sekarang dapat menentukan `--provision` alih-alih `--provision true` menginstal dengan penyediaan sumber daya otomatis. • Memperbaiki masalah saat perangkat inti tidak melaporkan statusnya ke layanan AWS IoT Greengrass cloud setelah penyediaan dalam skenario tertentu. • Peningkatan dan perbaikan kecil tambahan.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.4 pada 23 Maret 2022

Rilis ini menyediakan versi 2.5.4 komponen inti Greengrass dan versi 2.0.10 dari komponen peluncur Lambda.

Tanggal rilis: 23 Maret 2022

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.5.4 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Perbaikan bug umum dan perbaikan.
Peluncur Lambda	Versi 2.0.10 dari komponen peluncur Lambda tersedia. Perbaikan bug dan peningkatan • Perbaikan bug umum dan perbaikan.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.3 pada 6 Januari 2022

Rilis ini menyediakan versi 2.5.3 dari komponen inti Greengrass dan komponen penyedia PKCS #11 yang baru.

Tanggal rilis: 6 Januari 2022

Sorotan rilis

- Integrasi keamanan perangkat keras —Anda sekarang dapat mengonfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan kunci pribadi dan sertifikat yang Anda simpan dengan aman di modul keamanan perangkat keras (HSM). Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.5.3 dari inti Greengrass tersedia.

Komponen	Detail
	Fitur baru • Menambahkan dukungan untuk integrasi keamanan perangkat keras. Anda dapat menggunakan modul keamanan perangkat keras (HSM) untuk menyimpan kunci pribadi dan sertifikat perangkat dengan aman. Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras. Perbaikan bug dan peningkatan • Memperbaiki masalah dengan pengecualian runtime saat nukleus membuat koneksi MQTT dengan. AWS IoT Core
Penyedia PKCS #11	Versi 2.0.0 dari komponen penyedia PKCS #11 tersedia. Fitur baru • Menambahkan dukungan untuk integrasi keamanan perangkat keras. Anda dapat menggunakan modul keamanan perangkat keras (HSM) untuk menyimpan kunci pribadi dan sertifikat perangkat dengan aman. Lihat informasi yang lebih lengkap di Integrasi keamanan perangkat keras.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.2 pada 3 Desember 2021

Rilis ini menyediakan versi 2.5.2 dari komponen inti Greengrass.

Tanggal rilis: 3 Desember 2021

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.5.2 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana setelah pembaruan inti Greengrass, layanan Windows gagal memulai lagi setelah Anda menghentikannya atau me-reboot perangkat.
AWS IoT Device Defender	Versi 3.0.1 AWS IoT Device Defenderkomponen tersedia. Versi AWS IoT Device Defender komponen ini mengharapkan parameter konfigurasi yang berbeda dari versi 2.x. Jika Anda menggunakan konfigurasi non-default untuk versi 2.x, dan Anda ingin meningkatkan dari v2.x ke v3.x, Anda harus memperbarui konfigurasi komponen. Untuk informasi selengkap nya, lihat konfigurasi AWS IoT Device Defender komponen. Fitur baru • Menambahkan dukungan untuk perangkat inti yang menjalankan Windows.

Komponen	Detail
	• Mengubah jenis komponen dari komponen Lambda menjadi komponen generik. Komponen ini sekarang tidak lagi bergantung pada komponen router langganan lama untuk membuat langganan. • Menambahkan parameter `UseInstaller` konfigurasi baru yang memungkinkan Anda menonaktifkan skrip instalasi yang menginstal dependensi komponen secara opsional.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.1 pada 23 November 2021

Rilis ini menyediakan versi 2.5.1 dari komponen inti Greengrass.

Tanggal rilis: 23 November 2021

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan

untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.5.1 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Menambahkan dukungan untuk versi 32-bit dari Java Runtime Environment (JRE) pada Windows. • Mengubah perilaku penghapusan grup untuk perangkat inti yang AWS IoT kebijakannya tidak memberikan `greengrass:ListThingGroupsForCoreDevice` izin. Dengan versi ini, penerapan berlanjut, mencatat peringatan, dan tidak menghapus komponen saat Anda menghapus perangkat inti dari grup sesuatu. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat. • Memperbaiki masalah dengan variabel lingkungan sistem yang disediakan oleh inti Greengrass untuk proses komponen Greengrass. Anda sekarang dapat me-restart komponen untuk menggunakan variabel lingkungan sistem terbaru.

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.5.0 pada 12 November 2021

Rilis ini menyediakan versi 2.5.0 dari komponen inti Greengrass, komponen AWS baru yang disediakan, dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 12 November 2021

Sorotan rilis

- Dukungan perangkat Windows —Anda sekarang dapat menjalankan perangkat lunak AWS IoT Greengrass Core pada perangkat yang menjalankan sistem operasi Windows. Lihat informasi yang

lebih lengkap di Platform dan persyaratan yang didukung dan Kompatibilitas fitur Greengrass oleh sistem operasi.

- Perilaku penghapusan grup hal baru —Anda sekarang dapat menghapus perangkat inti dari grup benda untuk menghapus komponen grup benda itu di penerapan berikutnya ke perangkat itu.

  > **Important**
  >
  > Sebagai hasil dari perubahan ini, AWS IoT kebijakan perangkat inti harus memiliki `greengrass:ListThingGroupsForCoreDevice` izin. Jika Anda menggunakan penginstal perangkat lunak AWS IoT Greengrass inti untuk menyediakan sumber daya, AWS IoT kebijakan default mengizinkan`greengrass:*`, yang menyertakan izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.

- Dukungan keamanan perangkat keras —Anda sekarang dapat mengonfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan modul keamanan perangkat keras (HSM), sehingga Anda dapat menyimpan kunci pribadi dan sertifikat perangkat dengan aman. Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras.
- Dukungan proxy HTTPS —Anda sekarang dapat mengkonfigurasi perangkat lunak AWS IoT Greengrass Core untuk terhubung melalui proxy HTTPS. Untuk informasi selengkapnya, lihat Hubungkan pada port 443 atau melalui proksi jaringan.

Detail rilis

- Pembaruan dukungan platform
- Pembaruan komponen publik

Pembaruan dukungan platform

Platform	Detail
Windows	AWS IoT Greengrasssekarang mendukung menjalankan perangkat lunak AWS IoT Greengrass Core pada versi Windows berikut: • Windows 10 • Windows Server 2019

Platform	Detail
	Lihat informasi yang lebih lengkap di Platform dan persyaratan yang didukung dan Kompatibilitas fitur Greengrass oleh sistem operasi.

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.5.0 dari inti Greengrass tersedia. Fitur baru • Menambahkan dukungan untuk perangkat inti yang menjalankan Windows. • Ubah perilaku penghapusan grup benda. Dengan versi ini, Anda dapat menghapus perangkat inti dari grup benda untuk menghapus komponen grup benda itu di penerapan berikutnya.

Komponen	Detail
	Sebagai hasil dari perubahan ini, AWS IoT kebijakan perangkat inti harus memiliki `greengrass:ListThingGroupsForCoreDevice` izin. Jika Anda menggunakan penginstal perangkat lunak AWS IoT Greengrass inti untuk menyediakan sumber daya, AWS IoT kebijakan default mengizinkan`greengrass:*` , yang menyertakan izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass. • Menambahkan dukungan untuk konfigurasi proxy HTTPS. Untuk informasi selengkapnya, lihat Hubungkan pada port 443 atau melalui proksi jaringan. • Menambahkan parameter `windowsUser` konfigurasi baru. Anda dapat menggunakan parameter ini untuk menentukan pengguna default yang akan digunakan untuk menjalankan komponen pada perangkat inti Windows. Untuk informasi selengkapnya, lihat Konfigurasikan pengguna yang menjalankan komponen. • Menambahkan opsi `httpClient` konfigurasi baru yang dapat Anda gunakan untuk menyesuaikan batas waktu permintaan HTTP untuk meningkatkan kinerja pada jaringan yang lambat. Untuk informasi selengkapnya, lihat parameter konfigurasi HttpClient. Perbaikan bug dan peningkatan • Memperbaiki opsi siklus hidup bootstrap untuk me-restart perangkat inti dari komponen. • Menambahkan dukungan untuk tanda hubung dalam variabel resep. • Memperbaiki otorisasi IPC untuk komponen fungsi Lambda sesuai permintaan. • Meningkatkan pesan log dan mengubah log non-kritis dari `DEBUG` tingkat `INFO` ke tingkat, sehingga log lebih berguna. • Menghapus `iot:DescribeCertificate` izin dari peran pertukaran token default yang dibuat oleh inti Greengrass saat Anda menginstal AWS IoT Greengrass perangkat lunak Core dengan penyediaan otomatis. Izin ini tidak digunakan oleh inti Greengrass.

Komponen	Detail
	• Memperbaiki masalah sehingga skrip penyediaan otomatis tidak memerlukan `iam:GetPolicy` izin jika tersedia untuk kebijakan `iam:CreatePolicy` yang sama. • Peningkatan dan perbaikan kecil tambahan.
CLI Greengrass	Versi 2.5.0 dari CLI Greengrass tersedia. Fitur baru • Menambahkan dukungan untuk perangkat inti yang menjalankan Windows. • Menambahkan parameter `AuthorizedWindowsGroups` konfigurasi baru yang dapat Anda tentukan untuk mengotorisasi grup sistem untuk menggunakan CLI Greengrass pada perangkat Windows. • Menambahkan `windowsUser` parameter untuk penerapan lokal. Anda dapat menggunakan parameter ini menentukan pengguna yang akan digunakan untuk menjalankan komponen pada perangkat inti Windows.

Komponen	Detail
CloudWatch metrik	Versi 3.0.0 dari komponen CloudWatchmetrik tersedia. Versi komponen CloudWatch metrik ini mengharapkan parameter konfigurasi yang berbeda dari versi 2.x. Jika Anda menggunakan konfigurasi non-defau lt untuk versi 2.x, dan Anda ingin meningkatkan dari v2.x ke v3.x, Anda harus memperbarui konfigurasi komponen. Untuk informasi selengkapnya, lihat konfigurasi komponen CloudWatch metrik. Fitur baru • Menambahkan dukungan untuk perangkat inti yang menjalankan Windows. • Mengubah jenis komponen dari komponen Lambda menjadi komponen generik. Komponen ini sekarang tidak lagi bergantung pada komponen router langganan lama untuk membuat langganan. • Menambahkan parameter `InputTopic` konfigurasi baru untuk menentukan topik yang komponen berlangganan untuk menerima pesan. • Menambahkan parameter `OutputTopic` konfigurasi baru untuk menentukan topik yang komponen menerbitkan tanggapan status. • Menambahkan parameter `PubSubToIoTCore` konfigurasi baru untuk menentukan apakah akan mempublikasikan dan berlangganan topik AWS IoT Core MQTT. • Menambahkan parameter `UseInstaller` konfigurasi baru yang memungkinkan Anda menonaktifkan skrip instalasi yang menginstal dependensi komponen secara opsional. Perbaikan bug dan peningkatan Menambahkan dukungan untuk stempel waktu duplikat dalam data input.

Komponen	Detail
Manajer Lambda	Versi 2.2.0 dari komponen manajer Lambda tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana fungsi Lambda tidak dapat menulis log setelah restart. • Memperbaiki masalah saat router langganan lama mengirimkan pesan duplikat saat ada wildcard dalam topik tersebut. • Memperbaiki masalah di mana fungsi Lambda yang tidak disematkan tidak dapat menggunakan pustaka komunikasi antarproses Greengrass (IPC) di perpustakaan. AWS IoT Device SDK

Rilis: Pembaruan perangkat lunak AWS IoT Greengrass Core v2.4.0 pada 3 Agustus 2021

Rilis ini menyediakan versi 2.4.0 dari komponen inti Greengrass, komponen AWS baru yang disediakan, dan pembaruan ke komponen yang disediakan. AWS

Tanggal rilis: 3 Agustus 2021

Sorotan rilis

- Batas sumber daya sistem —Komponen inti Greengrass sekarang mendukung batas sumber daya sistem. Anda dapat mengonfigurasi jumlah maksimum penggunaan CPU dan RAM yang dapat digunakan oleh setiap proses komponen pada perangkat inti. Untuk informasi selengkapnya, lihat Konfigurasikan batas sumber daya sistem untuk komponen.
- Komponen jeda/lanjutkan —Inti Greengrass sekarang mendukung jeda dan melanjutkan komponen. Anda dapat menggunakan pustaka komunikasi antarproses (IPC) untuk mengembangkan komponen kustom yang menjeda dan melanjutkan proses komponen lain. Lihat informasi yang lebih lengkap di PauseComponent dan ResumeComponent.
- Instal dengan penyediaan AWS IoT armada —Gunakan plugin penyediaan AWS IoT armada baru untuk menginstal perangkat lunak AWS IoT Greengrass Inti pada perangkat yang terhubung ke AWS IoT sumber daya yang diperlukan penyediaan. AWS Perangkat menggunakan sertifikat klaim untuk penyediaan. Anda dapat menyematkan sertifikat klaim pada perangkat selama pembuatan, sehingga setiap perangkat dapat menyediakan segera setelah online. Untuk informasi

selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan AWS IoT armada.

- Instal dengan penyediaan khusus —Kembangkan plugin penyediaan khusus untuk menyediakan AWS sumber daya yang diperlukan saat Anda menginstal perangkat lunak Inti di AWS IoT Greengrass perangkat. Anda dapat membuat aplikasi Java yang berjalan selama instalasi untuk mengatur perangkat inti Greengrass untuk kasus penggunaan kustom Anda. Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya khusus.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.4.0 dari inti Greengrass tersedia.

Komponen	Detail
	Fitur baru • Menambahkan dukungan untuk batas sumber daya sistem. Anda dapat mengonfigurasi jumlah maksimum penggunaan CPU dan RAM yang dapat digunakan oleh setiap proses komponen pada perangkat inti. Untuk informasi selengkapnya, lihat Konfigurasikan batas sumber daya sistem untuk komponen. • Menambahkan operasi IPC untuk menjeda dan melanjutkan komponen. Lihat informasi yang lebih lengkap di PauseComponent dan ResumeComponent. • Menambahkan dukungan untuk penyediaan plugin. Anda dapat menentukan file JAR untuk dijalankan selama instalasi untuk menyediakan AWS sumber daya yang diperlukan untuk perangkat inti Greengrass. Inti Greengrass menyertakan antarmuka yang dapat Anda terapkan untuk mengembangkan plugin penyediaan khusus. Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya khusus. • Menambahkan `thing-name-policy` argumen opsional ke penginstal perangkat lunak AWS IoT Greengrass Core. Anda dapat menggunakan opsi ini untuk menentukan AWS IoT kebijakan yang ada atau kustom saat Anda menginstal perangkat lunak AWS IoT Greengrass Inti dengan penyediaan sumber daya otomatis. Perbaikan bug dan peningkatan • Memperbarui konfigurasi logging saat startup. Ini memperbaiki masalah di mana konfigurasi logging tidak diterapkan saat startup. • Memperbarui symlink pemuat nukleus untuk menunjuk ke penyimpanan komponen di folder root Greengrass selama penginstalan. Pembaruan ini memungkinkan Anda untuk menghapus file JAR dan artefak inti lainnya yang Anda unduh saat Anda menginstal perangkat lunak AWS IoT Greengrass Core. • Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.

Komponen	Detail
CLI Greengrass	Versi 2.4.0 dari CLI Greengrass tersedia. Fitur baru • Menambahkan dukungan untuk batas sumber daya sistem. Saat Anda membuat penyebaran lokal, Anda dapat mengonfigurasi jumlah maksimum penggunaan CPU dan RAM yang dapat digunakan oleh setiap proses komponen pada perangkat inti. Untuk informasi selengkap nya, lihat Konfigurasikan batas sumber daya sistem untuk komponen dan perintah deployment create.
AWS IoTpenyed iaan armada dengan klaim	Penyediaan AWS IoT armada dengan plugin klaim sekarang tersedia. Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan AWS IoT armada. Fitur baru • Menambahkan dukungan untuk menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan AWS IoT armada. Selama instalasi, perangkat terhubung AWS IoT ke penyediaan AWS sumber daya yang diperlukan dan mengunduh sertifikat perangkat untuk digunakan untuk operasi reguler.

Rilis: Pembaruan perangkat lunak inti v2.3.0 AWS IoT Greengrass pada 29 Juni 2021

Rilis ini menyediakan versi 2.3.0 dari komponen inti Greengrass.

Tanggal rilis: 29 Juni 2021

Sorotan rilis

- Dukungan konfigurasi besar—Komponen nukleus Greengrass sekarang mendukung deployment dokumen hingga 10 MB. Anda sekarang dapat men-deploy pembaruan konfigurasi yang lebih besar untuk komponen Greengrass.

> **Note**
>
> Untuk menggunakan fitur ini, kebijakan AWS IoT harus memungkinkan izin `greengrass:GetDeploymentConfiguration`. Jika Anda menggunakan penginstal perangkat lunak inti AWS IoT Greengrass untuk menyediakan sumber daya, kebijakan AWS IoT perangkat inti Anda akan memungkinkan `greengrass:*`, yang mencakup izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.3.0 dari inti Greengrass tersedia.

Komponen	Detail
	Fitur baru • Menambahkan dukungan untuk dokumen konfigurasi deployment hingga 10 MB, naik dari 7 KB (untuk deployment yang menargetkan objek) atau 31 KB (untuk deployment yang menargetkan objek grup). Untuk menggunakan fitur ini, kebijakan AWS IoT harus memungkinkan izin `greengrass:GetDeploymentConfiguration` . Jika Anda menggunakan penginstal perangkat lunak inti AWS IoT Greengrass untuk menyediakan sumber daya, kebijakan AWS IoT perangkat inti Anda akan memungkinkan `greengrass:*` , yang mencakup izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass. • Menambahkan variabel resep `iot:thingName` . Anda dapat menggunakan variabel resep ini untuk mendapatkan nama objek AWS IoT perangkat inti dalam resep. Untuk informasi selengkapnya, lihat Variabel resep. Perbaikan bug dan peningkatan • Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.

Rilis: Pembaruan perangkat lunak inti v2.2.0 AWS IoT Greengrass pada tanggal 18 Juni 2021

Rilis ini menyediakan versi 2.2.0 dari komponen inti Greengrass, yang disediakan oleh AWS yang baru, dan pembaruan pada komponen yang disediakan oleh AWS.

Tanggal rilis: 18 Juni 2021

Sorotan rilis

- Dukungan perangkat klien—Komponen perangkat klien yang yang disediakan oleh AWS yang baru memungkinkan Anda untuk menghubungkan perangkat klien ke perangkat inti Anda dengan menggunakan penemuan cloud. Anda dapat menyinkronkan perangkat klien dengan AWS IoT

Core dan berinteraksi dengan perangkat klien pada komponen Greengrass. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

- Layanan bayangan lokal—Komponen shadow manager memungkinkan layanan bayangan lokal pada perangkat inti Anda. Anda dapat menggunakan layanan bayangan ini untuk berinteraksi dengan bayangan lokal saat offline dengan menggunakan pustaka komunikasi antarproses (IPC) Greengrass di AWS IoT Device SDK. Anda juga dapat menggunakan komponen shadow manager untuk menyinkronkan perangkat keadaan bayangan lokal dengan AWS IoT Core. Untuk informasi selengkapnya, lihat Berinteraksilah dengan bayangan perangkat.

Detail rilis

- Pembaruan komponen publik

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.2.0 dari inti Greengrass kini tersedia.

Komponen	Detail
	Fitur baru • Menambahkan operasi IPC untuk manajemen bayangan lokal. Perbaikan bug dan peningkatan • Mengurangi ukuran file JAR. • Mengurangi penggunaan memori. • Memperbaiki masalah di mana konfigurasi log tidak diperbarui dalam kasus tertentu. • Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.
Manajer bayangan	Versi 2.0.0 dari komponen shadow manager yang baru kini tersedia. Fitur baru • Menambahkan dukungan untuk bayangan klasik dan bernama. • Menambahkan dukungan untuk manajemen bayangan lokal dengan menggunakan IPC. • Menambahkan dukungan untuk sinkronisasi bayangan dengan AWS IoT Core.
Autentikasi perangkat klien	Versi 2.0.0 dari komponen auth perangkat klien yang baru kini tersedia. Fitur baru • Menambahkan dukungan untuk perangkat klien Greengrass, yang merupakan perangkat IoT lokal yang terhubung ke perangkat inti melalui MQTT. • Menambahkan dukungan untuk autentikasi dan otorisasi perangkat klien dan tindakan MQTT-nya.
Pialang MQTT Moquette	Versi 2.0.0 dari komponen broker MQTT Moquette yang baru kini tersedia. Fitur baru • Menambahkan dukungan untuk broker MQTT Moquette lokal yang menangani komunikasi dengan perangkat klien.

Komponen	Detail
Jembatan MQTT	Versi 2.0.0 dari komponen jembatan MQTT Moquette yang baru kini tersedia. Fitur baru • Menambahkan dukungan untuk merelai pesan antara broker MQTT lokal, broker publikasi/berlangganan Greengrass lokal, dan broker MQTT AWS IoT Core.
Detektor IP	Versi 2.0.0 dari komponen detektor IP yang baru kini tersedia. Fitur baru • Menambahkan dukungan untuk melaporkan titik akhir broker MQTT lokal perangkat inti ke layanan cloud AWS IoT Greengrass agar perangkat klien dapat terhubung.
Manajer log	Versi 2.1.1 dari komponen manajer log kini tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah saat konfigurasi log sistem tidak diperbarui dalam kasus tertentu.
Deteksi objek DLR	Versi 2.1.2 dari deteksi objek DLR kini tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah penskalaan gambar yang mengakibatkan kotak batas yang tidak akurat dalam hasil inferensi deteksi objek DLR sampel.
TensorFlow Deteksi objek Lite	Versi 2.1.1 dari deteksi objek TensorFlow Lite tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah penskalaan gambar yang mengakibatkan kotak pembatas tidak akurat dalam hasil inferensi deteksi objek TensorFlow Lite sampel.

Rilis: Pembaruan perangkat lunak inti v2.1.0 AWS IoT Greengrass pada tanggal 26 April 2021

Rilis ini menyediakan versi 2.1.0 dari komponen inti Greengrass dan memperbarui komponen yang disediakan oleh AWS.

Tanggal rilis: 26 April 2021

Sorotan rilis

- Integrasi Docker Hub dan Amazon Elastic Container Registry (Amazon ECR)—Komponen manajer aplikasi Docker yang baru memungkinkan Anda mengunduh gambar publik atau privat dari Amazon ECR. Anda juga dapat menggunakan komponen ini untuk mengunduh gambar publik dari Docker Hub dan AWS Marketplace. Untuk informasi selengkapnya, lihat Jalankan kontainer Docker.
- Dockerfile dan gambar Docker untuk Perangkat lunak inti AWS IoT Greengrass—Anda dapat menggunakan gambar Docker Greengrass untuk menjalankan AWS IoT Greengrass dalam kontainer Docker yang menggunakan Amazon Linux 2 sebagai sistem operasi dasar. Anda juga dapat menggunakan Dockerfile AWS IoT Greengrass untuk membangun citra Greengrass Anda sendiri. Untuk informasi selengkapnya, lihat Jalankan perangkat lunak AWS IoT Greengrass Core dalam wadah Docker.
- Dukungan untuk kerangka kerja dan platform pembelajaran mesin tambahan —Anda dapat menerapkan komponen inferensi pembelajaran mesin sampel yang menggunakan model yang telah dilatih sebelumnya untuk melakukan klasifikasi gambar sampel dan deteksi objek menggunakan TensorFlow Lite 2.5.0 dan DLR 1.6.0. Rilis ini juga memperluas dukungan sampel machine learning untuk perangkat Armv8 (AArch64). Untuk informasi selengkapnya, lihat Lakukan inferensi machine learning.

Detail rilis

- Pembaruan dukungan platform
- Pembaruan komponen publik

Pembaruan dukungan platform

Platform	Detail
Docker	Dockerfile dan gambar Docker untuk AWS IoT Greengrass sekarang tersedia. Dockerfile AWS IoT Greengrass menyediakan Dockerfile untuk membangun sebuah gambar kontainer yang memiliki perangkat lunak inti AWS IoT Greengrass dan dependensi yang diinstal pada gambar dasar Amazon Linux 2 (x86_64). Anda dapat memodifikasi gambar dasar di Dockerfile untuk menjalankan AWS IoT Greengrass pada arsitektur platform yang berbeda. Gambar Docker AWS IoT Greengrass menyediakan gambar Docker bawaan yang memiliki perangkat lunak inti AWS IoT Greengrass dan dependensi yang diinstal pada gambar dasar Amazon Linux 2 (x86_64). Untuk informasi selengkapnya, lihat Jalankan perangkat lunak AWS IoT Greengrass Core dalam wadah Docker.

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung

saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail

Inti Greengrass	Versi 2.1.0 dari inti Greengrass tersedia.
	
	Fitur baru
	• Mendukung pengunduhan gambar Docker dari repositori privat di Amazon ECR.
	• Menambahkan parameter berikut untuk menyesuaikan konfigurasi MQTT pada perangkat inti:
	  • `maxInFlightPublishes` - Jumlah maksimum pesan MQTT QoS 1 yang tidak diakui yang dapat terbang pada waktu yang sama.
	  • `maxPublishRetry` - Jumlah waktu maksimum untuk mencoba kembali pesan yang gagal untuk dipublikasikan.
	• Menambahkan parameter konfigurasi `fleetstatusservice` untuk mengonfigurasi interval di mana perangkat inti menerbitkan status perangkat untuk AWS Cloud.
	• Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.
	
	Perbaikan bug dan peningkatan
	• Memperbaiki masalah yang menyebabkan deployment bayangan diduplikasi saat nukleus dimulai ulang.
	• Memperbaiki masalah yang menyebabkan nukleus lumpuh saat mengalami pengecualian beban layanan.
	• Meningkatkan resolusi dependensi komponen untuk menggagalkan deployment yang mencakup dependensi melingkar.
	• Memperbaiki masalah yang mencegah komponen plugin di-deploy ulang jika komponen tersebut sebelumnya telah dihapus dari perangkat inti.
	• Memperbaiki masalah yang menyebabkan variabel lingkungan `HOME` yang akan ditetapkan ke direktori */greengrass/v2* `/work` untuk

Komponen	Detail
	komponen Lambda atau untuk komponen yang berjalan sebagai root. Variabel `HOME` sekarang diatur dengan tepat ke direktori home untuk pengguna yang menjalankan komponen. • Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.
Manajer aplikasi Docker	Versi 2.0.0 dari komponen pengelola aplikasi docker yang baru tersedia. Fitur baru • Mengelola kredensial untuk men-download gambar dari repositori privat di Amazon ECR. • Mengunduh gambar publik dari Amazon ECR, Docker Hub, dan AWS Marketplace.
Peluncur Lambda	Versi 2.0.4 dari komponen peluncur Lambda kini tersedia. Perbaikan bug dan peningkatan • Memperbaiki masalah saat komponen tidak meneruskan dengan tepat `AddGroupOwner` ke kontainer fungsi Lambda.
Router langganan lama	Versi 2.1.0 dari komponen router langganan warisan kini tersedia. Perbaikan bug dan peningkatan • Menambahkan dukungan untuk menentukan nama komponen dan bukan ARN untuk `source` dan `target`. Jika Anda menentukan nama komponen untuk suatu langganan, Anda tidak perlu mengonfigurasi ulang langganan setiap kali versi fungsi Lambda berubah.
Konsol debug lokal	Versi 2.1.0 dari komponen konsol debug lokal kini tersedia. Fitur baru • Menggunakan HTTPS untuk mengamankan koneksi Anda ke konsol debug lokal. HTTPS tidak diaktifkan secara default. Perbaikan bug dan peningkatan • Anda dapat mengabaikan pesan flashbar di editor konfigurasi.

Komponen	Detail
Manajer log	Versi 2.1.0 dari komponen manajer log kini tersedia. Perbaikan bug dan peningkatan • Gunakan default untuk `logFileDirectoryPath` dan `logFileRegex` yang bekerja untuk komponen Greengrass yang mencetak ke output standar (stdout) dan kesalahan standar (stderr). • Rutekan lalu lintas dengan benar melalui proxy jaringan yang dikonfigurasi saat mengunggah CloudWatch log ke Log. • Tangani karakter titik dua (`:`) dengan benar di nama aliran log. CloudWatch Nama aliran log log tidak mendukung titik dua. • Sederhanakan nama aliran log dengan menghapus nama grup objek dari aliran log. • Hapus pesan log kesalahan yang dicetak selama perilaku normal.

Komponen	Detail
Klasifikasi gambar DLR	Versi 2.1.1 dari komponen klasifikasi gambar DLR kini tersedia. Fitur baru • Gunakan Deep Learning Runtime v1.6.0. • Tambahkan dukungan untuk klasifikasi gambar sampel pada platform Armv8 (AArch64). Hal ini akan memperluas dukungan machine learning untuk perangkat inti Greengrass yang menjalankan NVIDIA Jetson, seperti Jetson Nano. • Aktifkan integrasi kamera untuk inferensi sampel. Gunakan parameter konfigurasi `UseCamera` yang baru untuk mengaktifkan kode kesimpulan sampel untuk mengakses kamera pada perangkat inti Greengrass Anda dan jalankan inferensi lokal pada gambar yang ditangkap. • Tambahkan dukungan untuk menerbitkan hasil inferensi ke AWS Cloud. Gunakan parameter konfigurasi `PublishResultsOnTopic` yang baru untuk menentukan topik di mana Anda ingin mempublikasikan hasil. • Tambahkan parameter konfigurasi `ImageDirectory` yang baru yang memungkinkan Anda untuk menentukan direktori kustom untuk gambar di mana Anda ingin melakukan inferensi. Perbaikan bug dan peningkatan • Tulis hasil inferensi ke file log komponen dan bukan file inferensi terpisah. • Gunakan modul pencatatan perangkat lunak inti AWS IoT Greengrass untuk mencatat output komponen. • Gunakan AWS IoT Device SDK untuk membaca konfigurasi komponen dan menerapkan perubahan konfigurasi.

Komponen	Detail
Deteksi objek DLR	Versi 2.1.1 dari komponen deteksi objek DLR kini tersedia. Fitur baru • Gunakan Deep Learning Runtime v1.6.0. • Tambahkan dukungan untuk deteksi objek sampel pada platform Armv8 (AArch64). Hal ini akan memperluas dukungan machine learning untuk perangkat inti Greengrass yang menjalankan NVIDIA Jetson, seperti Jetson Nano. • Aktifkan integrasi kamera untuk inferensi sampel. Gunakan parameter konfigurasi `UseCamera` yang baru untuk mengaktifkan kode kesimpulan sampel untuk mengakses kamera pada perangkat inti Greengrass Anda dan jalankan inferensi lokal pada gambar yang ditangkap. • Tambahkan dukungan untuk menerbitkan hasil inferensi ke AWS Cloud. Gunakan parameter konfigurasi `PublishResultsOnTopic` yang baru untuk menentukan topik di mana Anda ingin mempublikasikan hasil. • Tambahkan parameter konfigurasi `ImageDirectory` yang baru yang memungkinkan Anda untuk menentukan direktori kustom untuk gambar di mana Anda ingin melakukan inferensi. Perbaikan bug dan peningkatan • Tulis hasil inferensi ke file log komponen dan bukan file inferensi terpisah. • Gunakan modul pencatatan perangkat lunak inti AWS IoT Greengrass untuk mencatat output komponen. • Gunakan AWS IoT Device SDK untuk membaca konfigurasi komponen dan menerapkan perubahan konfigurasi.

Komponen	Detail
Toko model klasifikasi gambar DLR	Versi 2.1.1 dari komponen penyimpanan model klasifikasi gambar DLR kini tersedia. Fitur baru • Tambahkan contoh model klasifikasi gambar ResNet -50 untuk platform Armv8 (AArch64). Hal ini akan memperluas dukungan machine learning untuk perangkat inti Greengrass yang menjalankan NVIDIA Jetson, seperti Jetson Nano.
Toko model deteksi objek DLR	Versi 2.1.1 dari komponen penyimpanan model deteksi objek DLR kini tersedia. Fitur baru • Tambahkan sampel model deteksi objek YOLOv3 pada platform Armv8 (AArch64). Hal ini akan memperluas dukungan machine learning untuk perangkat inti Greengrass yang menjalankan NVIDIA Jetson, seperti Jetson Nano.
Pemasang DLR	Versi 1.6.1 dari komponen DLR kini tersedia. Fitur baru • Instal Deep Learning Runtime v1.6.0 dan dependensinya. • Tambahkan dukungan untuk menginstal DLR pada platform Armv8 (AArch64). Hal ini akan memperluas dukungan machine learning untuk perangkat inti Greengrass yang menjalankan NVIDIA Jetson, seperti Jetson Nano. Perbaikan bug dan peningkatan • Instal AWS IoT Device SDK di lingkungan virtual untuk membaca konfigurasi komponen dan menerapkan perubahan konfigurasi. • Peningkatan dan perbaikan kecil tambahan.

Komponen	Detail
TensorFlow Klasifikasi gambar ringan	Versi 2.1.0 komponen klasifikasi gambar TensorFlow Lite baru tersedia. Fitur baru • Tambahkan dukungan untuk inferensi klasifikasi gambar sampel menggunakan TensorFlow Lite.
TensorFlow Deteksi objek ringan	Versi 2.1.0 dari komponen deteksi objek TensorFlow Lite baru tersedia. Fitur baru • Tambahkan dukungan untuk inferensi deteksi objek sampel menggunakan TensorFlow Lite.
TensorFlow Toko model klasifikasi gambar Lite	Versi 2.1.0 komponen penyimpanan model klasifikasi gambar TensorFlow Lite baru tersedia. Fitur baru • Berikan model terkuantisasi MobileNet v1 yang telah dilatih sebelumnya untuk inferensi klasifikasi gambar sampel menggunakan Lite. TensorFlow
TensorFlow Toko model deteksi objek Lite	Versi 2.1.0 dari komponen penyimpanan model deteksi objek TensorFlow Lite baru tersedia. Fitur baru • Menyediakan MobileNet model Single Shot Detection (SSD) pra-terlatih yang dilatih pada dataset COCO untuk inferensi deteksi objek sampel menggunakan Lite. TensorFlow
TensorFlow Lite	Versi 2.5.0 dari komponen TensorFlow Lite baru tersedia. Fitur baru • Instal TensorFlow Lite v1.6.0 dan dependensinya di lingkungan virtual pada platform Armv7, Armv8 (AArch64), dan x86_64.

Rilis: Pembaruan perangkat lunak Core v2.0.5 AWS IoT Greengrass pada 09 Maret 2021

Rilis ini menyediakan versi 2.0.5 dari komponen inti Greengrass dan pembaruan komponen yang disediakan oleh AWS. Rilis ini memperbaiki masalah dengan dukungan proksi jaringan dan masalah dengan titik akhir bidang data Greengrass di Wilayah Cina AWS.

Tanggal rilis: 09 Maret 2021

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

> **Important**
>
> Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
> Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.0.5 dari inti Greengrass tersedia. Perbaikan bug dan peningkatan • Dengan benar mengarahkan lalu lintas melalui proksi jaringan yang dikonfigurasi saat mengunduh komponen yang disediakan oleh AWS.

Komponen	Detail
	• Menggunakan titik akhir bidang data Greengrass yang benar di Wilayah Cina AWS.

Rilis: Pembaruan perangkat lunak inti v2.0.4 AWS IoT Greengrass pada tanggal 04 Februari 2021

Rilis ini menyediakan versi 2.0.4 dari komponen inti Greengrass. Termasuk di dalamnya parameter `greengrassDataPlanePort` yang baru untuk mengonfigurasi komunikasi HTTPS melalui port 443 dan memperbaiki bug. Kebijakan IAM minimal sekarang memerlukan `iam:GetPolicy` dan `sts:GetCallerIdentity` saat penginstal perangkat lunak inti AWS IoT Greengrass dijalankan dengan `--provision true`.

Tanggal rilis: 04 Februari 2021

Pembaruan komponen publik

Tabel berikut ini mencantumkan komponen yang disediakan oleh AWS yang mencakup fitur baru dan diperbarui.

Important

Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Detail
Inti Greengrass	Versi 2.0.4 dari inti Greengrass tersedia. Fitur baru • Memungkinkan lalu lintas HTTPS melalui port 443. Anda dapat menggunakan parameter konfigurasi `greengrassDataPlanePort` untuk versi 2.0.4 komponen inti untuk mengonfigurasi komunikasi HTTPS untuk berjalan melalui port 443 dan bukan port default 8443. Untuk informasi selengkapnya, lihat Konfigurasikan HTTPS melalui port 443. • Menambahkan variabel resep jalur kerja. Anda dapat menggunakan variabel resep ini untuk mendapatkan jalur ke folder kerja komponen, yang dapat Anda gunakan untuk berbagi file antara komponen dan dependensinya. Untuk informasi lebih lanjut, lihat variabel resep jalur kerja. Perbaikan bug dan peningkatan • Mencegah terciptanya kebijakan (IAM) role AWS Identity and Access Management pertukaran token jika kebijakan peran sudah ada. Akibat dari perubahan ini, penginstal sekarang memerlukan `iam:GetPolicy` dan `sts:GetCallerIdentity` ketika dijalankan dengan `--provision true` . Untuk informasi selengkapnya, lihat Kebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya. • Secara tepat menangani pembatalan deployment yang belum berhasil didaftarkan. • Meng-update konfigurasi untuk menghapus entri lama dengan cap waktu yang lebih baru ketika memutar kembali deployment. • Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.

Migrasi dari AWS IoT Greengrass Versi 1

AWS IoT Greengrass Version 2adalah rilis versi utama dari perangkat lunak AWS IoT Greengrass Core, API, dan konsol. AWS IoT Greengrass V2memperkenalkan beberapa perbaikanAWS IoT Greengrass V1, seperti aplikasi modular, penyebaran ke armada perangkat besar, dan dukungan untuk platform tambahan.

> **Note**
>
> Setelah 30 Juni 2023 AWS IoT Greengrass Version 1 tidak lagi menerima pembaruan fitur, penyempurnaan, perbaikan bug, atau tambalan keamanan. Untuk informasi selengkapnya, lihat kebijakan AWS IoT Greengrass V1 pemeliharaan. Jika Anda menggunakanAWS IoT Greengrass V1, kami sangat menyarankan Anda bermigrasi keAWS IoT Greengrass V2.

Ikuti petunjuk dalam panduan ini untuk bermigrasi dari AWS IoT Greengrass V1 keAWS IoT Greengrass V2.

Bisakah saya menjalankan aplikasi V1 saya di V2?

Sebagian besar aplikasi V1 dapat berjalan pada perangkat inti V2 tanpa perlu mengubah kode aplikasi. Jika aplikasi V1 Anda menggunakan fitur berikut, Anda tidak akan dapat menjalankannya di V2.

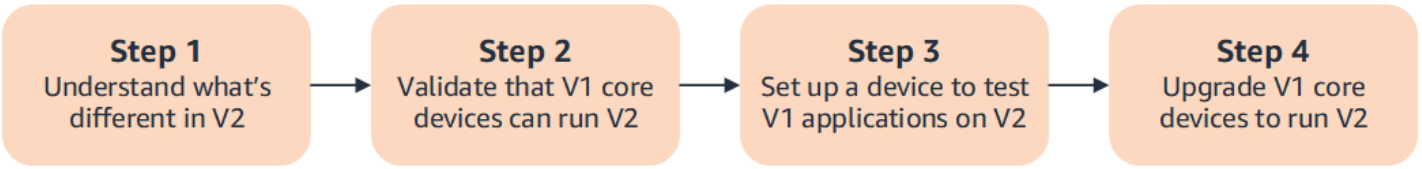
- Runtime fungsi C dan C++ Lambda

Jika aplikasi V1 Anda menggunakan salah satu dari fitur berikut, Anda harus memodifikasi kode aplikasi Anda untuk menggunakan AWS IoT Device SDK V2 untuk menjalankan aplikasi. AWS IoT Greengrass V2

- Berinteraksi dengan layanan bayangan lokal
- Publikasikan pesan ke perangkat lokal yang terhubung (perangkat Greengrass)

Ikhtisar migrasi

Pada tingkat tinggi, Anda dapat menggunakan prosedur berikut untuk meningkatkan perangkat inti dari AWS IoT Greengrass V1 keAWS IoT Greengrass V2. Prosedur pasti yang Anda ikuti tergantung pada persyaratan spesifik untuk lingkungan Anda.



1. Memahami perbedaan antara V1 dan V2

   AWS IoT Greengrass V2memperkenalkan konsep dasar baru untuk armada perangkat dan perangkat lunak yang dapat digunakan, dan V2 menyederhanakan beberapa konsep dari V1.

   Layanan AWS IoT Greengrass V2 cloud dan perangkat lunak AWS IoT Greengrass Core v2.x tidak kompatibel ke belakang dengan layanan AWS IoT Greengrass V1 cloud dan perangkat lunak AWS IoT Greengrass Core v1.x. Akibatnya, pembaruan AWS IoT Greengrass V1 over-the-air (OTA) tidak dapat memutakhirkan perangkat inti dari V1 ke V2.

2. Validasi bahwa perangkat inti V1 dapat menjalankan V2

   Validasi bahwa perangkat inti V1 dapat menjalankan perangkat lunak AWS IoT Greengrass Core v2.x dan fitur. AWS IoT Greengrass V2 AWS IoT Greengrass V2memiliki persyaratan perangkat yang berbeda dariAWS IoT Greengrass V1.

3. Siapkan perangkat baru untuk menguji aplikasi V1 di V2

   Untuk meminimalkan risiko pada perangkat Anda dalam produksi, buat perangkat baru untuk menguji aplikasi V1 Anda di V2. Setelah menginstal perangkat lunak AWS IoT Greengrass Core v2.x, Anda dapat membuat dan menerapkan AWS IoT Greengrass V2 komponen untuk memigrasi dan menguji aplikasi Anda. AWS IoT Greengrass V1

4. Tingkatkan perangkat inti V1 untuk menjalankan V2

   Tingkatkan perangkat inti V1 yang ada untuk menjalankan perangkat lunak AWS IoT Greengrass Core v2.x dan komponen. AWS IoT Greengrass V2 Untuk memigrasikan armada perangkat dari V1 ke V2, Anda ulangi langkah ini untuk setiap perangkat di armada.

Perbedaan antara AWS IoT Greengrass V1 dan AWS IoT Greengrass V2

AWS IoT Greengrass V2 memperkenalkan konsep dasar baru untuk perangkat, armada, dan perangkat lunak yang dapat digunakan. Bagian ini menjelaskan konsep V1 yang berbeda di V2.

Konsep dan terminologi Greengrass

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Kode aplikasi	Dalam AWS IoT Greengrass V1, fungsi Lambda mendefinisikan perangkat lunak yang berjalan pada perangkat inti. Dalam setiap grup Greengrass, Anda menentukan langganan dan sumber daya lokal yang menggunakan fungsi tersebut. Untuk fungsi Lambda yang dijalankan oleh perangkat lunak AWS IoT Greengrass Core di lingkungan runtime Lambda dalam kontainer, Anda menentukan parameter kontainer, seperti batas memori.	Dalam AWS IoT Greengrass V2, komponen adalah modul perangkat lunak yang berjalan pada perangkat inti. • Setiap komponen memiliki resep yang mendefinisikan metadata, parameter, dependensi, dan skrip komponen untuk dijalankan pada setiap langkah dalam siklus hidup komponen. • Resep ini juga mendefinisikan artefak komponen, yang merupakan file biner, seperti skrip, kode yang dikompilasi, dan sumber daya statis. • Ketika Anda men-deploy komponen ke perangkat inti, perangkat inti akan mengunduh resep komponen dan artefak untuk menjalankan komponen. Anda dapat mengimpor fungsi Lambda V1 Anda sebagai

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		komponen yang berjalan di lingkungan runtime Lambda di. AWS IoT Greengrass V2 Ketika Anda mengimpor fungsi Lambda, Anda menentuka n langganan, sumber daya lokal, dan parameter kontainer untuk fungsi tersebut. Untuk informasi selengkapnya, lihat Langkah 2: Membuat dan menyebarkan AWS IoT Greengrass V2 komponen untuk memigrasi aplikasi AWS IoT Greengrass V1. Untuk informasi selengkap nya tentang cara membuat komponen kustom, lihatKembangkan AWS IoT Greengrass komponen.

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
AWS IoT Greengrass grup dan penyebaran	Dalam AWS IoT Greengrass V1, grup mendefinisikan perangkat inti, pengaturan dan perangkat lunak untuk perangkat inti itu, dan daftar AWS IoT hal-hal yang dapat terhubung ke perangkat inti itu. Anda membuat penerapan untuk mengirim konfigurasi grup ke perangkat inti.	Di AWS IoT Greengrass V2, Anda menggunakan penerapan untuk menentukan komponen perangkat lunak dan konfigurasi yang berjalan pada perangkat inti. • Setiap penyebaran menargetkan perangkat inti tunggal (yang merupakan AWS IoT sesuatu) atau grup AWS IoT benda yang dapat berisi beberapa perangkat inti. • Penerapan ke grup benda bersifat kontinu, jadi ketika Anda menambahkan perangkat inti ke grup sesuatu, ia menerima konfigurasi perangkat lunak untuk grup itu. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat. Di AWS IoT Greengrass V2, Anda juga dapat membuat penerapan lokal menggunakan CLI Greengrass untuk menguji komponen perangkat lunak khusus pada perangkat tempat Anda mengembangkannya. Untuk informasi

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		selengkapnya, lihat Buat AWS IoT Greengrass komponen.
AWS IoT Greengrass Perangkat lunak inti	Di AWS IoT Greengrass V1, perangkat lunak AWS IoT Greengrass Core adalah paket tunggal yang berisi perangkat lunak dan semua fitur-fiturnya. Perangkat tepi tempat Anda menginstal perangkat lunak AWS IoT Greengrass Core disebut inti Greengrass.	Di AWS IoT Greengrass V2, perangkat lunak AWS IoT Greengrass Core bersifat modular, sehingga Anda dapat memilih apa yang akan diinstal untuk mengontrol jejak memori. • Komponen inti Greengrass adalah instalasi minimum yang diperlukan dari perangkat lunak Core. AWS IoT Greengrass Perangkat tepi tempat Anda memasang nukleus disebut perangkat inti Greengrass. • Nukleus menangani penerapan, orkestrasi, dan manajemen siklus hidup komponen lain pada perangkat inti. • Fitur seperti manajer aliran, manajer rahasia, dan pengelola log adalah komponen yang Anda gunakan hanya ketika Anda membutuhkan fitur tersebut. Untuk informasi selengkapnya, lihat Komponen yang disediakan oleh AWS.

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Konektor	Di AWS IoT Greengrass V1, konektor adalah modul bawaan yang Anda terapkan ke perangkat AWS IoT Greengrass V1 inti untuk berinteraksi dengan infrastruktur lokal, protokol perangkat AWS, dan layanan cloud lainnya.	Di AWS IoT Greengrass V2, AWS menyediakan komponen Greengrass yang mengimplementasikan fungsionalitas yang disediakan oleh konektor di V1. AWS IoT Greengrass V2 Komponen-komponen berikut menyediakan fungsionalitas konektor Greengrass V1: • CloudWatch komponen metrik • AWS IoT Device Defender komponen • Komponen Firehose • Komponen adaptor protokol Modbus-RTU • Komponen Amazon SNS Untuk informasi selengkapnya, lihat Komponen yang disediakan oleh AWS.

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Perangkat yang terhubung (perangkat Greengrass)	Di AWS IoT Greengrass V1, perangkat yang terhubung adalah AWS IoT hal-hal yang Anda tambahkan ke grup Greengrass untuk terhubung ke perangkat inti dalam grup itu dan berkomunikasi melalui MQTT. Anda harus menggunakan grup tersebut setiap kali menambahkan atau menghapus perangkat yang tersambung. Anda menggunakan langganan untuk menyampaikan pesan antara perangkat yang terhubung AWS IoT Core, dan aplikasi pada perangkat inti.	Di AWS IoT Greengrass V2, perangkat yang terhubung disebut perangkat klien Greengrass. • Anda mengaitkan perangkat klien ke perangkat inti untuk menghubungkannya dan berkomunikasi melalui MQTT. • Untuk mengotorisasi perangkat klien agar terhubung, Anda menentukan kebijakan otorisasi yang dapat diterapkan ke grup perangkat klien, sehingga Anda tidak perlu membuat penerapan untuk menambah atau menghapus perangkat klien. • Untuk menyampaikan pesan antara perangkat klien AWS IoT Core, dan komponen Greengrass, Anda dapat mengonfigurasi komponen jembatan MQTT opsional. Di keduanya AWS IoT Greengrass V1 dan AWS IoT Greengrass V2, perangkat dapat menjalankan FreeRTOS atau menggunakan AWS IoT Device SDKatau Greengrass discovery API untuk

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		mendapatkan informasi tentang perangkat inti yang dapat mereka sambungkan. Greengrass discovery API kompatibel ke belakang, jadi jika Anda memiliki perangkat klien yang terhubung ke perangkat inti V1, Anda dapat menghubungkannya ke perangkat inti V2 tanpa mengubah kodenya. Untuk informasi selengkap nya tentang perangkat klien, lihatBerinteraksilah dengan perangkat IoT lokal.

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Sumber daya lokal	Di AWS IoT Greengrass V1, fungsi Lambda yang berjalan dalam kontainer dapat dikonfigurasi untuk mengakses volume dan perangkat pada sistem file perangkat inti. Sumber daya sistem file ini dikenal sebagai sumber daya lokal.	Di AWS IoT Greengrass V2, Anda dapat menjalankan komponen yang merupakan fungsi Lambda, kontainer Docker, atau proses sistem operasi asli atau runtime kustom. • Saat Anda mengimpor fungsi Lambda dalam kontainer sebagai komponen, Anda harus menentukan sumber daya lokal yang digunakan fungsi tersebut. • Fungsi Lambda non-kontainer dan komponen non-Lambda dapat bekerja secara langsung dengan sumber daya lokal pada perangkat inti, jadi Anda tidak perlu menentukan sumber daya lokal yang digunakan komponen tersebut.

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Layanan bayangan lokal	Di AWS IoT Greengrass V1, layanan bayangan lokal diaktifkan secara default, dan hanya mendukung bayangan klasik yang tidak disebutkan namanya. Anda menggunak an AWS IoT Greengrass Core SDK di fungsi Lambda Anda untuk berinteraksi dengan bayangan di perangkat Anda.	Di AWS IoT Greengrass V2, Anda mengaktifkan layanan bayangan lokal dengan menerapkan komponen manajer bayangan. • Anda dapat menggunak an AWS IoT Device SDK V2 dalam fungsi Lambda dan komponen khusus untuk berinteraksi dengan bayangan di perangkat Anda. • Layanan bayangan lokal mendukung bayangan bernama. • Layanan bayangan lokal memungkinkan Anda menghapus bayangan dan menyinkronkan bayangan yang dihapus dengan AWS IoT Core. Untuk informasi selengkapnya, lihat Berinteraksilah dengan bayangan perangkat.

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Langganan	Di AWS IoT Greengrass V1, Anda menentukan langganan untuk grup Greengrass untuk menentukan saluran komunikasi antara fungsi Lambda, konektor, perangkat yang terhubung, broker MQTT, dan layanan bayangan lokal AWS IoT Core . Langganan menentukan di mana fungsi Lambda menerima pesan peristiwa untuk digunakan sebagai muatan fungsi.	Di AWS IoT Greengrass V2, Anda menentukan saluran komunikasi tanpa menggunakan langganan. • Komponen mengelola saluran komunikasi mereka sendiri untuk berinteraksi dengan pesan penerbitan/berlangganan lokal, pesan AWS IoT Core MQTT, dan layanan bayangan lokal. • Untuk mengembangkan komponen yang bereaksi terhadap pesan dari komponen lain atau broker AWS IoT Core MQTT, Anda dapat menggunakan antarmuka komunikasi antarproses (IPC) untuk pesan penerbitan/berlangganan lokal dan pesan MQTT. AWS IoT Core • Untuk mengembangkan komponen yang berinteraksi dengan layanan bayangan lokal, Anda dapat menggunakan antarmuka IPC untuk layanan bayangan lokal. • Dalam konfigurasi komponen, Anda menentukan kebijakan

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		otorisasi untuk menentukan topik dan bayangan lokal yang diizinkan oleh komponen untuk digunakan. • Untuk mengonfigurasi saluran komunikasi antara perangkat klien, broker penerbitan/berlangganan lokal, dan broker AWS IoT Core MQTT, Anda mengonfigurasi dan menerapkan komponen jembatan MQTT. Komponen jembatan MQTT memungkinkan Anda berinteraksi dengan perangkat klien dalam komponen dan menyampaikan pesan antara perangkat klien dan perangkat klien. AWS IoT Core

Konsep	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Mengakses lainnya Layanan AWS	Di AWS IoT Greengras s V1, Anda melampirkan peran AWS Identity and Access Management (IAM), yang disebut peran grup, ke grup Greengrass. Peran grup menentukan izin yang digunakan oleh fungsi dan AWS IoT Greengrass fitur Lambda pada perangkat inti grup tersebut untuk mengakses. Layanan AWS	Di AWS IoT Greengrass V2, Anda melampirkan alias AWS IoT peran ke perangkat inti Greengrass. Alias peran menunjuk ke peran IAM yang disebut peran pertukaran token. Peran pertukaran token mendefinisikan izin yang digunakan komponen Greengrass pada perangkat inti untuk mengakses. Layanan AWS Lihat informasi yang lebih lengkap di Otorisasi perangkat inti untuk berintera ksi dengan AWS layanan.

Validasi perangkat inti V1 dapat menjalankan perangkat lunak V2

Perangkat lunakAWS IoT Greengrass Core v2.x memiliki persyaratan yang berbeda dari v1.x perangkat lunakAWS IoT Greengrass Core. Sebelum Anda meningkatkan perangkat inti V1 ke V2, tinjau persyaratan perangkat untukAWS IoT Greengrass V2. AWS IoT Greengrass V2saat ini tidak mendukung migrasi untuk sistem berbasis Linux kustom menggunakan Proyek Yocto.

Anda dapat menggunakan AWS IoT Device Tester(IDT)AWS IoT Greengrass V2 untuk memvalidasi bahwa perangkat memenuhi persyaratan untuk menjalankan perangkat lunakAWS IoT Greengrass Core v2.x. IDT adalah kerangka pengujian download yang berjalan pada komputer host Anda dan terhubung ke perangkat yang akan divalidasi. Ikuti petunjuk untuk menggunakan IDT untuk menjalankanAWS IoT Greengrass kualifikasi suite. Saat mengonfigurasi IDT, Anda dapat memilih untuk memvalidasi apakah perangkat mendukung fitur opsional, seperti Docker, machine learning (MS), manajemen aliran data, dan integrasi keamanan perangkat keras.

Jika IDT melaporkan kegagalan pengujian V2 atau kesalahan untuk perangkat inti V1, Anda tidak dapat meningkatkan perangkat tersebut dari V1 ke V2.

Siapkan perangkat inti V2 baru untuk menguji aplikasi V1

Siapkan perangkat AWS IoT Greengrass V2 inti baru untuk menyebarkan dan menguji komponen dan AWS Lambda fungsi AWS yang disediakan untuk aplikasi Anda AWS IoT Greengrass V1 . Anda juga dapat menggunakan perangkat inti V2 ini untuk mengembangkan dan menguji komponen Greengrass kustom tambahan yang menjalankan proses asli pada perangkat inti. Setelah menguji aplikasi pada perangkat inti V2, Anda dapat meningkatkan perangkat inti V1 yang ada ke V2 dan menerapkan komponen V2 yang menyediakan fungsionalitas V1 Anda.

Langkah 1: Instal AWS IoT Greengrass V2 di perangkat baru

Instal perangkat lunak AWS IoT Greengrass Core v2.x pada perangkat baru. Anda dapat mengikuti tutorial memulai untuk menyiapkan perangkat dan mempelajari cara mengembangkan dan menyebarkan komponen. Tutorial ini menggunakan penyediaan otomatis untuk mengatur perangkat dengan cepat. Saat Anda menginstal perangkat lunak AWS IoT Greengrass Core v2.x, tentukan `--deploy-dev-tools` argumen untuk menerapkan CLI Greengrass, sehingga Anda dapat mengembangkan, menguji, dan men-debug komponen langsung di perangkat. Untuk informasi selengkapnya tentang opsi penginstalan lainnya, termasuk cara menginstal perangkat lunak AWS IoT Greengrass inti di belakang proxy atau menggunakan modul keamanan perangkat keras (HSM), lihatInstal perangkat lunak inti AWS IoT Greengrass.

(Opsional) Aktifkan logging ke Amazon CloudWatch Logs

Untuk mengaktifkan perangkat inti V2 mengunggah log ke Amazon CloudWatch Logs, Anda dapat menerapkan komponen pengelola log AWS yang disediakan. Anda dapat menggunakan CloudWatch Log untuk melihat log komponen, sehingga Anda dapat men-debug dan memecahkan masalah tanpa akses ke sistem file perangkat inti. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Langkah 2: Membuat dan menyebarkan AWS IoT Greengrass V2 komponen untuk memigrasi aplikasi AWS IoT Greengrass V1

Anda dapat menjalankan sebagian besar AWS IoT Greengrass V1 aplikasi di AWS IoT Greengrass V2. Anda dapat mengimpor fungsi Lambda sebagai komponen yang berjalan AWS IoT Greengrass V2, dan Anda dapat menggunakan komponen yang AWS disediakan yang menawarkan fungsionalitas yang sama dengan konektor. AWS IoT Greengrass

Anda juga dapat mengembangkan komponen khusus untuk membangun fitur atau runtime apa pun untuk dijalankan di perangkat inti Greengrass. Untuk informasi tentang cara mengembangkan dan menguji komponen secara lokal, lihatBuat AWS IoT Greengrass komponen.

Topik

- Impor fungsi Lambda V1
- Gunakan konektor V1
- Jalankan kontainer Docker
- Jalankan inferensi machine learning
- Hubungkan perangkat Greengrass V1
- Aktifkan layanan bayangan lokal
- Integrasikan dengan AWS IoT SiteWise

Impor fungsi Lambda V1

Anda dapat mengimpor fungsi Lambda sebagai AWS IoT Greengrass V2 komponen. Pilih dari pendekatan berikut:

- Impor V1 Lambda berfungsi langsung sebagai komponen Greengrass.
- Perbarui fungsi Lambda Anda untuk menggunakan pustaka Greengrass di v2 AWS IoT Device SDK , lalu impor fungsi Lambda sebagai komponen Greengrass.
- Buat komponen khusus yang menggunakan kode non-Lambda dan AWS IoT Device SDK v2 untuk mengimplementasikan fungsionalitas yang sama dengan fungsi Lambda Anda.

Jika fungsi Lambda Anda menggunakan fitur, seperti pengelola aliran atau rahasia lokal, Anda harus menentukan dependensi pada komponen yang AWS disediakan yang mengemas fitur ini. Ketika Anda menggunakan komponen fungsi Lambda, deployment tersebut juga mencakup komponen untuk setiap fitur yang Anda tetapkan sebagai dependensi. Dalam penerapan, Anda dapat mengonfigurasi parameter, seperti rahasia mana yang akan diterapkan ke perangkat inti. Tidak semua fitur V1 memerlukan dependensi komponen untuk fungsi Lambda Anda pada V2. Daftar berikut menjelaskan cara menggunakan fitur V1 di komponen fungsi Lambda V2 Anda.

- Akses AWS layanan lain

  Jika fungsi Lambda Anda menggunakan AWS kredensional untuk membuat permintaan ke AWS layanan lain, peran pertukaran token perangkat inti harus mengizinkan perangkat inti untuk

melakukan operasi AWS yang digunakan fungsi Lambda. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

- Manajer aliran

  Jika fungsi Lambda Anda menggunakan stream manager, tentukan `aws.greengrass.StreamManager` sebagai dependensi komponen ketika Anda mengimpor fungsi tersebut. Saat Anda menggunakan komponen stream manager, tentukan parameter stream manager yang akan ditetapkan untuk perangkat inti target. Peran pertukaran token perangkat inti harus memungkinkan perangkat inti mengakses AWS Cloud tujuan yang Anda gunakan dengan pengelola aliran. Untuk informasi selengkapnya, lihat Manajer pengaliran.

- Rahasia lokal

  Jika fungsi Lambda Anda menggunakan rahasia lokal, tentukan `aws.greengrass.SecretManager` sebagai dependensi komponen ketika Anda mengimpor fungsi tersebut. Ketika Anda men-deploy komponen secret manager, tentukan sumber daya rahasia yang akan di-deploy ke perangkat inti target. Peran pertukaran token perangkat inti harus memungkinkan perangkat inti untuk mengambil sumber daya rahasia untuk digunakan. Untuk informasi selengkapnya, lihat Secrets manager.

  Saat Anda menerapkan komponen fungsi Lambda Anda, konfigurasikan agar memiliki kebijakan otorisasi IPC yang memberikan izin untuk menggunakan operasi IPC GetSecretValue di V2. AWS IoT Device SDK

- Bayangan lokal

  Jika fungsi Lambda Anda berinteraksi dengan bayangan lokal, Anda harus memperbarui kode fungsi Lambda untuk menggunakan V2. AWS IoT Device SDK Anda juga harus menentukan `aws.greengrass.ShadowManager` sebagai dependensi komponen ketika Anda mengimpor fungsi tersebut. Untuk informasi selengkapnya, lihat Berinteraksilah dengan bayangan perangkat.

  Saat Anda menerapkan komponen fungsi Lambda Anda, konfigurasikan agar memiliki kebijakan otorisasi IPC yang memberikan izin untuk menggunakan operasi IPC bayangan di V2. AWS IoT Device SDK

- Langganan
  - Jika fungsi Lambda Anda berlangganan pesan dari sumber cloud, tentukan langganan tersebut sebagai sumber peristiwa saat Anda mengimpor fungsi tersebut.
  - Jika fungsi Lambda Anda berlangganan pesan dari fungsi Lambda lain, atau jika fungsi Lambda Anda menerbitkan pesan ke atau fungsi Lambda AWS IoT Core lainnya, konfigurasikan dan

terapkan komponen router langganan lama saat Anda menerapkan fungsi Lambda Anda. Saat Anda men-deploy komponen router langganan warisan, tentukan langganan yang digunakan oleh fungsi Lambda tersebut.

> **Note**
>
> Komponen router langganan lama hanya diperlukan jika fungsi Lambda Anda menggunakan fungsi `publish()` di Core AWS IoT Greengrass SDK. Jika Anda memperbarui kode fungsi Lambda Anda untuk menggunakan antarmuka komunikasi antarproses (IPC) di AWS IoT Device SDK V2, Anda tidak perlu menggunakan komponen router langganan lama. Untuk informasi lebih lanjut, lihat layanan komunikasi antar proses berikut ini:
>
> - Pesan lokal publikasi/berlangganan
> - Terbitkan/berlangganan pesan MQTT AWS IoT Core

- Jika fungsi Lambda Anda berlangganan pesan dari perangkat lokal yang terhubung, tentukan langganan tersebut sebagai sumber peristiwa saat Anda mengimpor fungsi tersebut. Anda juga harus mengonfigurasi dan menerapkan komponen jembatan MQTT untuk menyampaikan pesan dari perangkat yang terhubung ke topik penerbitan/langganan lokal yang Anda tentukan sebagai sumber peristiwa.
- Jika fungsi Lambda Anda menerbitkan pesan ke perangkat lokal yang terhubung, Anda harus memperbarui kode fungsi Lambda untuk menggunakan AWS IoT Device SDK V2 untuk mempublikasikan pesan penerbitan/berlangganan lokal. Anda juga harus mengonfigurasi dan menerapkan komponen jembatan MQTT untuk menyampaikan pesan dari broker pesan penerbitan/berlangganan lokal ke perangkat yang terhubung.

- Volume dan perangkat lokal

  Jika fungsi Lambda terkontainerisasi Anda mengakses volume atau perangkat lokal, tentukan volume dan perangkat tersebut saat Anda mengimpor fungsi Lambda. Fitur ini tidak memerlukan dependensi komponen.

Untuk informasi selengkapnya, lihat Jalankan fungsi AWS Lambda.

Gunakan konektor V1

Anda dapat menerapkan komponen yang AWS disediakan yang menawarkan fungsionalitas yang sama dari beberapa AWS IoT Greengrass konektor. Bila Anda membuat deployment, Anda dapat mengonfigurasi parameter konektor.

AWS IoT Greengrass V2 Komponen-komponen berikut menyediakan fungsionalitas konektor Greengrass V1:

- CloudWatch komponen metrik
- AWS IoT Device Defender komponen
- Komponen Firehose
- Komponen adaptor protokol Modbus-RTU
- Komponen Amazon SNS

Jalankan kontainer Docker

AWS IoT Greengrass V2 tidak menyediakan komponen untuk secara langsung mengganti konektor penerapan aplikasi V1 Docker. Namun, Anda dapat menggunakan komponen pengelola aplikasi Docker untuk mengunduh gambar Docker, dan kemudian membuat komponen khusus yang menjalankan kontainer Docker dari gambar yang diunduh. Lihat informasi yang lebih lengkap di Jalankan kontainer Docker dan Manajer aplikasi Docker.

Jalankan inferensi machine learning

AWS IoT Greengrass V2 menyediakan komponen Amazon SageMaker Edge Manager yang menginstal agen Amazon SageMaker Edge Manager dan memungkinkan Anda menggunakan model yang SageMaker dikompilasi NEO sebagai komponen model pada perangkat inti Greengrass. AWS IoT Greengrass V2 juga menyediakan komponen yang menginstal Deep Learning Runtime dan TensorFlow Lite di perangkat Anda. Anda dapat menggunakan model DLR dan TensorFlow Lite serta komponen inferensi yang sesuai untuk melakukan klasifikasi gambar sampel dan inferensi deteksi objek. Untuk menggunakan kerangka kerja pembelajaran mesin lainnya, seperti MXNet TensorFlow dan, Anda dapat mengembangkan komponen kustom Anda sendiri yang menggunakan kerangka kerja ini.

Hubungkan perangkat Greengrass V1

Perangkat yang AWS IoT Greengrass V1 terhubung di disebut perangkat klien di AWS IoT Greengrass V2. AWS IoT Greengrass V2 dukungan untuk perangkat klien kompatibel ke belakang AWS IoT Greengrass V1, sehingga Anda dapat menghubungkan perangkat klien V1 ke perangkat inti V2 tanpa mengubah kode aplikasinya. Untuk mengaktifkan perangkat klien terhubung ke perangkat inti V2, gunakan komponen Greengrass yang memungkinkan dukungan perangkat klien, dan kaitkan perangkat klien ke perangkat inti. Untuk merelai pesan antara perangkat klien, layanan cloud AWS IoT Core , dan komponen Greengrass (termasuk fungsi Lambda), deploy dan konfigurasikan Komponen jembatan MQTT. Anda dapat men-deploy komponen detektor IP untuk secara otomatis mendeteksi informasi konektivitas, atau Anda dapat mengelola titik akhir secara manual. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Aktifkan layanan bayangan lokal

Di AWS IoT Greengrass V2, layanan bayangan lokal diimplementasikan oleh komponen manajer bayangan AWS yang disediakan. AWS IoT Greengrass V2 juga termasuk dukungan untuk bayangan bernama. Untuk mengaktifkan komponen Anda berinteraksi dengan bayangan lokal dan menyinkronkan status bayangan AWS IoT Core, konfigurasikan dan terapkan komponen pengelola bayangan, dan gunakan operasi IPC bayangan dalam kode komponen Anda. Untuk informasi selengkapnya, lihat Berinteraksilah dengan bayangan perangkat.

Integrasikan dengan AWS IoT SiteWise

Jika Anda menggunakan perangkat inti V1 sebagai AWS IoT SiteWise gateway, ikuti petunjuk untuk menyiapkan perangkat inti V2 baru Anda sebagai AWS IoT SiteWise gateway. AWS IoT SiteWise menyediakan skrip instalasi yang menyebarkan AWS IoT SiteWise komponen untuk Anda.

Langkah 3: Uji AWS IoT Greengrass V2 aplikasi Anda

Setelah Anda membuat dan menerapkan komponen V2 ke perangkat inti V2 baru Anda, verifikasi bahwa aplikasi Anda memenuhi harapan Anda. Anda dapat memeriksa log perangkat untuk melihat pesan keluaran standar (stdout) dan kesalahan standar (stderr) komponen Anda. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Jika Anda menerapkan CLI Greengrass ke perangkat inti, Anda dapat menggunakannya untuk men-debug komponen dan konfigurasinya. Untuk informasi selengkapnya, lihat Perintah Greengrass CLI.

Setelah Anda memverifikasi bahwa aplikasi Anda bekerja pada perangkat inti V2, Anda dapat menerapkan komponen Greengrass aplikasi Anda ke perangkat inti lainnya. Jika Anda

mengembangkan komponen kustom yang menjalankan proses asli atau kontainer Docker, Anda harus terlebih dahulu mempublikasikan komponen tersebut ke AWS IoT Greengrass layanan untuk menerapkannya ke perangkat inti lainnya.

Tingkatkan perangkat inti Greengrass V1 ke Greengrass V2

Setelah memverifikasi bahwa aplikasi dan komponen berfungsi pada perangkat AWS IoT Greengrass V2 inti, Anda dapat menginstal perangkat lunak AWS IoT Greengrass Core v2.x di perangkat yang saat ini menjalankan v1.x, seperti perangkat produksi. Kemudian, gunakan komponen Greengrass V2 untuk menjalankan aplikasi Greengrass Anda di perangkat.

Untuk meningkatkan armada perangkat dari V1 ke V2, selesaikan langkah-langkah ini agar setiap perangkat dapat ditingkatkan. Anda dapat menggunakan grup benda untuk menyebarkan komponen V2 ke armada perangkat inti.

> **Tip**
>
> Kami menyarankan Anda membuat skrip untuk mengotomatiskan proses peningkatan untuk armada perangkat. Jika Anda menggunakannya AWS Systems Manageruntuk mengelola armada, Anda dapat menggunakan Systems Manager untuk menjalankan skrip tersebut di setiap perangkat untuk meningkatkan armada Anda dari V1 ke V2.
> Anda dapat menghubungi perwakilan Support AWS Enterprise Anda untuk mengajukan pertanyaan tentang cara terbaik untuk mengotomatisasi proses upgrade.

Langkah 1: Instal perangkat lunak AWS IoT Greengrass Core v2.x

Pilih dari opsi berikut untuk menginstal perangkat lunak AWS IoT Greengrass Core v2.x pada perangkat inti V1:

- Tingkatkan dalam langkah yang lebih sedikit

  Untuk meningkatkan dalam langkah yang lebih sedikit, Anda dapat menghapus instalasi perangkat lunak v1.x sebelum Anda menginstal perangkat lunak v2.x.
- Upgrade dengan downtime minimal

  Untuk meningkatkan dengan downtime minimal, Anda dapat menginstal kedua versi perangkat lunak AWS IoT Greengrass Core secara bersamaan. Setelah Anda menginstal perangkat lunak

AWS IoT Greengrass Core v2.x dan memverifikasi bahwa aplikasi V2 Anda beroperasi dengan benar, Anda menghapus instalasi perangkat lunak AWS IoT Greengrass Core v1.x. Sebelum Anda memilih opsi ini, pertimbangkan RAM tambahan yang diperlukan untuk menjalankan kedua versi perangkat lunak AWS IoT Greengrass Core secara bersamaan.

Copot pemasangan AWS IoT Greengrass Core v1.x sebelum Anda menginstal v2.x

Jika Anda ingin memutakhirkan secara berurutan, hapus instalan perangkat lunak AWS IoT Greengrass Core v1.x sebelum Anda menginstal v2.x di perangkat Anda.

Untuk menghapus instalasi perangkat lunak AWS IoT Greengrass Core v1.x

1. Jika perangkat lunak AWS IoT Greengrass Core v1.x berjalan sebagai layanan, Anda harus menghentikan, menonaktifkan, dan menghapus layanan.

   a. Hentikan layanan v1.x perangkat lunak AWS IoT Greengrass Core yang sedang berjalan.

   ```
   sudo systemctl stop greengrass
   ```

   b. Tunggu sampai layanan berhenti. Anda dapat menggunakan `list` perintah untuk memeriksa status layanan.

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

   c. Nonaktifkan layanan.

   ```
   sudo systemctl disable greengrass
   ```

   d. Hapus layanan.

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

2. Jika perangkat lunak AWS IoT Greengrass Core v1.x tidak berjalan sebagai layanan, gunakan perintah berikut untuk menghentikan daemon. Ganti *`greengrass-root dengan nama folder root`* Greengrass Anda. Lokasi default adalah `/greengrass`.

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

3. (Opsional) Cadangkan folder root Greengrass Anda dan, jika ada, folder tulis khusus Anda, ke folder lain di perangkat Anda.

   a. Gunakan perintah berikut untuk menyalin folder root Greengrass saat ini ke folder lain, lalu hapus folder root.

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

   b. Gunakan perintah berikut untuk memindahkan folder tulis ke folder lain, lalu hapus folder tulis.

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

Anda kemudian dapat menggunakan petunjuk penginstalan AWS IoT Greengrass V2 untuk menginstal perangkat lunak pada perangkat Anda.

> **Tip**
>
> Untuk menggunakan kembali identitas perangkat inti saat Anda memigrasikannya dari V1 ke V2, ikuti petunjuk untuk menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan manual. Pertama-tama hapus perangkat lunak inti V1 dari perangkat, lalu gunakan kembali AWS IoT barang dan sertifikat perangkat inti V1, dan perbarui AWS IoT kebijakan sertifikat untuk memberikan izin yang dibutuhkan perangkat lunak v2.x.

Instal perangkat lunak AWS IoT Greengrass Core v2.x pada perangkat yang sudah menjalankan v1.x

Jika Anda menginstal perangkat lunak AWS IoT Greengrass Core v2.x pada perangkat yang sudah menjalankan perangkat lunak AWS IoT Greengrass Core v1.x, ingatlah hal berikut:

- Nama AWS IoT benda untuk perangkat inti V2 Anda harus unik. Jangan gunakan nama yang sama dengan perangkat inti V1 Anda.
- Port yang Anda gunakan untuk perangkat lunak AWS IoT Greengrass Core v2.x harus berbeda dari port yang Anda gunakan untuk v1.x.

  - Konfigurasikan manajer aliran V1 untuk menggunakan port selain 8088. Untuk informasi selengkapnya, lihat Mengonfigurasi pengelola aliran.
  - Konfigurasikan broker V1 MQTT untuk menggunakan port selain 8883. Untuk informasi selengkapnya, lihat Mengkonfigurasi port MQTT untuk pesan lokal.
- AWS IoT Greengrass V2tidak menyediakan opsi untuk mengganti nama layanan sistem Greengrass. Jika Anda menjalankan Greengrass sebagai layanan sistem, Anda harus melakukan salah satu hal berikut untuk menghindari nama layanan sistem yang bertentangan:
  - Ganti nama layanan Greengrass untuk v1.x sebelum Anda menginstal v2.x.
  - Instal perangkat lunak AWS IoT Greengrass Core v2.x tanpa layanan sistem, lalu konfigurasikan perangkat lunak secara manual sebagai layanan sistem dengan nama selain. `greengrass`

  Untuk mengganti nama layanan Greengrass untuk v1.x

  1. Hentikan layanan v1.x perangkat lunak AWS IoT Greengrass Core.

     ```
     sudo systemctl stop greengrass
     ```

  2. Tunggu sampai layanan berhenti. Layanan ini dapat memakan waktu hingga beberapa menit untuk berhenti. Anda dapat menggunakan `list-units` perintah untuk memeriksa apakah layanan berhenti.

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

  3. Nonaktifkan layanan.

     ```
     sudo systemctl disable greengrass
     ```

  4. Ganti nama layanan.

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

  5. Muat ulang layanan dan mulai.

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

Anda kemudian dapat menggunakan petunjuk penginstalan AWS IoT Greengrass V2 untuk menginstal perangkat lunak pada perangkat Anda.

> **Tip**
>
> Untuk menggunakan kembali identitas perangkat inti saat Anda memigrasikannya dari V1 ke V2, ikuti petunjuk untuk menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan manual. Pertama-tama hapus perangkat lunak inti V1 dari perangkat, lalu gunakan kembali AWS IoT barang dan sertifikat perangkat inti V1, dan perbarui AWS IoT kebijakan sertifikat untuk memberikan izin yang dibutuhkan perangkat lunak v2.x.

Langkah 2: Menyebarkan AWS IoT Greengrass V2 komponen ke perangkat inti

Setelah Anda menginstal perangkat lunak AWS IoT Greengrass Core v2.x di perangkat Anda, buat penerapan yang menyertakan sumber daya berikut. Untuk menyebarkan komponen ke armada perangkat serupa, buat penyebaran untuk grup benda yang berisi perangkat tersebut.

- Komponen fungsi Lambda yang Anda buat dari fungsi Lambda V1 Anda. Untuk informasi selengkapnya, lihat Jalankan fungsi AWS Lambda.
- Jika Anda menggunakan langganan V1, komponen router langganan lama.
- Jika Anda menggunakan stream manager, komponen stream manager. Untuk informasi selengkapnya, lihat Kelola aliran data di perangkat inti Greengrass.
- Jika Anda menggunakan rahasia lokal, komponen manajer rahasia.
- Jika Anda menggunakan konektor V1, komponen konektor AWSyang disediakan.
- Jika Anda menggunakan kontainer Docker, komponen manajer aplikasi Docker. Untuk informasi selengkapnya, lihat Jalankan kontainer Docker.
- Jika Anda menggunakan inferensi pembelajaran mesin, komponen untuk dukungan pembelajaran mesin. Untuk informasi selengkapnya, lihat Lakukan inferensi machine learning.
- Jika Anda menggunakan perangkat yang terhubung, komponen untuk dukungan perangkat klien. Anda juga harus mengaktifkan dukungan perangkat klien dan mengaitkan perangkat klien dengan perangkat inti Anda. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.
- Jika Anda menggunakan bayangan perangkat, komponen pengelola bayangan. Untuk informasi selengkapnya, lihat Berinteraksilah dengan bayangan perangkat.

- Jika Anda mengunggah log dari perangkat inti Greengrass ke CloudWatch Amazon Logs, komponen pengelola log. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.
- Jika Anda mengintegrasikan denganAWS IoT SiteWise, ikuti petunjuk untuk mengatur perangkat inti V2 sebagai AWS IoT SiteWise gateway. AWS IoT SiteWisemenyediakan skrip instalasi yang menyebarkan AWS IoT SiteWise komponen untuk Anda.
- Komponen yang ditentukan pengguna yang Anda kembangkan untuk mengimplementasikan fungsionalitas khusus.

Untuk informasi tentang membuat dan merevisi penerapan, lihat. Deploy komponen AWS IoT Greengrass ke perangkat

Tutorial: Memulai dengan AWS IoT Greengrass V2

Anda dapat menyelesaikan tutorial memulai ini untuk mempelajari fitur-fitur dasarAWS IoT Greengrass V2. Dalam tutorial ini, Anda melakukan hal-hal berikut:

1. Instal dan konfigurasikan perangkat lunak AWS IoT Greengrass Core pada perangkat Linux, seperti Raspberry Pi, atau perangkat Windows. Perangkat ini adalah perangkat inti Greengrass.
2. Mengembangkan komponen Hello World pada perangkat inti Greengrass Anda. Komponen adalah modul perangkat lunak yang berjalan pada perangkat inti Greengrass.
3. Unggah komponen tersebut ke AWS IoT Greengrass V2 pada AWS Cloud.
4. Deploy komponen tersebut dari AWS Cloud ke perangkat inti Greengrass Anda.

> **Note**
>
> Tutorial ini menjelaskan cara mengatur lingkungan pengembangan dan mengeksplorasi fitur AWS IoT Greengrass. Untuk informasi selengkapnya tentang cara mengatur dan mengonfigurasi perangkat produksi, lihat berikut ini:
>
> - Menyiapkan perangkat AWS IoT Greengrass inti
> - Instal perangkat lunak inti AWS IoT Greengrass

Anda dapat mengharapkan untuk menghabiskan 20 hingga 30 menit pada tutorial ini.

Topik

Prasyarat

Untuk menyelesaikan tutorial memulai ini, Anda memerlukan hal berikut ini:

- Sesi Akun AWS. Jika Anda tidak memilikinya, lihat Langkah 1: Siapkan AWS akun.
- Penggunaan sebuah Wilayah AWSyang mendukungAWS IoT Greengrass V2. Untuk daftar Wilayah yang didukung, lihat AWS IoT Greengrass V2titik akhir dan kuota di. Referensi Umum AWS
- Pengguna (IAM) AWS Identity and Access Management dengan izin administrator.
- Perangkat untuk diatur sebagai perangkat inti Greengrass, seperti Raspberry Pi dengan Raspberry Pi OS (sebelumnya disebut Raspbian), atau perangkat Windows 10. Anda harus memiliki izin administrator pada perangkat ini, atau kemampuan untuk memperoleh hak administrator, seperti melalui. `sudo` Perangkat ini harus memiliki koneksi internet.

  Anda juga dapat memilih untuk menggunakan perangkat lain yang memenuhi persyaratan untuk menginstal dan menjalankan perangkat lunak AWS IoT Greengrass Core. Untuk informasi selengkapnya, lihat Platform dan persyaratan yang didukung.

  Jika komputer pengembangan Anda memenuhi persyaratan ini, Anda dapat mengaturnya sebagai perangkat inti Greengrass Anda dalam tutorial ini.
- Python 3.5 atau yang lebih baru diinstal untuk semua pengguna di perangkat dan ditambahkan ke variabel `PATH` lingkungan. Di Windows, Anda juga harus menginstal Python Launcher untuk Windows untuk semua pengguna.

  Important

  Di Windows, Python tidak menginstal untuk semua pengguna secara default. Ketika Anda menginstal Python, Anda harus menyesuaikan instalasi untuk mengkonfigurasinya untuk perangkat lunak AWS IoT Greengrass Core untuk menjalankan skrip Python. Misalnya, jika Anda menggunakan penginstal Python grafis, lakukan hal berikut:

  1. Pilih Instal peluncur untuk semua pengguna (disarankan).
  2. Pilih Customize installation.
  3. Pilih Next.
  4. Pilih Install for all users.
  5. Pilih Add Python to environment variables.

6. Pilih Instal.

Untuk informasi selengkapnya, lihat Menggunakan Python di Windows dalam dokumentasi Python 3.

- AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi dengan kredensial di komputer pengembangan dan perangkat Anda. Pastikan Anda menggunakan Wilayah AWS yang sama untuk mengonfigurasi AWS CLI di komputer pengembangan dan perangkat Anda. Untuk menggunakannya AWS IoT Greengrass V2AWS CLI, Anda harus memiliki salah satu versi berikut atau yang lebih baru:
  - Minimum AWS CLI Versi V1: v1.18.197
  - Minimum AWS CLI Versi V2: v2.1.11

> **Tip**
>
> Anda dapat menjalankan perintah berikut untuk memeriksa versi AWS CLI yang Anda miliki.
>
> ```
> aws --version
> ```

Untuk informasi selengkapnya, lihat Menginstal, memperbarui, dan melepas pemasangan AWS CLI dan Mengonfigurasi AWS CLI di Panduan Pengguna AWS Command Line Interface.

> **Note**
>
> Jika Anda menggunakan perangkat ARM 32-bit, seperti Raspberry Pi dengan sistem operasi 32-bit, instal AWS CLI V1. AWS CLI V2 tidak tersedia untuk perangkat ARM 32-bit. Untuk informasi selengkapnya, lihat Menginstal, memperbarui, dan mencopot instalasi AWS CLI versi 1.

Langkah 1: Siapkan AWS akun

Mendaftar untuk Akun AWS

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

Untuk mendaftar untuk Akun AWS

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

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

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

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

Buat pengguna dengan akses administratif

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

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

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

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

Buat pengguna dengan akses 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 pengguna.

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

Masuk sebagai pengguna dengan akses administratif

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

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

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.
2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Langkah 2: Siapkan lingkungan Anda

Ikuti langkah-langkah di bagian ini untuk menyiapkan perangkat Linux atau Windows untuk digunakan sebagai perangkat AWS IoT Greengrass inti Anda.

Siapkan perangkat Linux (Raspberry Pi)

Langkah-langkah ini menganggap bahwa Anda menggunakan Raspberry Pi dengan Raspberry Pi OS. Jika Anda menggunakan perangkat atau sistem operasi yang berbeda, baca dokumentasi yang relevan untuk perangkat Anda.

Untuk mengatur Raspberry Pi untuk AWS IoT Greengrass V2

1. Aktifkan SSH pada Anda Raspberry Pi untuk terhubung secara jarak jauh dengannya. Untuk informasi selengkapnya, lihat, SSH (Secure shell) dalam Dokumentasi Raspberry Pi.
2. Temukan alamat IP Raspberry Pi Anda untuk terhubung ke sana dengan SSH. Untuk melakukannya, Anda dapat menjalankan perintah berikut pada Raspberry Pi Anda.

   ```
   hostname -I
   ```

3. Hubungkan ke Raspberry Pi Anda dengan SSH.

   Pada komputer pengembangan Anda, jalankan perintah berikut. Ganti *nama* pengguna dengan nama pengguna untuk masuk, dan ganti *pi-ip-address*dengan alamat IP yang Anda temukan di langkah sebelumnya.

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

   > **Important**
   >
   > Jika komputer pengembangan Anda menggunakan versi Windows yang lebih lama, Anda mungkin tidak memiliki perintah `ssh`, atau Anda mungkin memiliki `ssh` tetapi tidak dapat terhubung ke Raspberry Pi Anda. Untuk terhubung ke Raspberry Pi Anda, Anda dapat menginstal dan mengonfigurasi PuTTY, yang merupakan klien SSH tanpa biaya dan open source. Baca Dokumentasi PuTTY untuk terhubung ke Raspberry Pi Anda..

4. Pasang waktu aktif Java, yang diperlukan oleh perangkat lunak inti AWS IoT Greengrass agar berjalan. Pada Raspberry Pi Anda, gunakan perintah berikut untuk menginstal Java 11.

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

   Ketika instalasi selesai, jalankan perintah berikut untuk memverifikasi bahwa Java berjalan pada Raspberry Pi Anda.

```
java -version
```

Perintah mencetak versi Java yang berjalan pada perangkat. Output-nya akan terlihat serupa dengan yang berikut ini.

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

Tip: Atur parameter kernel pada Raspberry Pi

Jika perangkat Anda adalah Raspberry Pi, Anda dapat menyelesaikan langkah-langkah berikut untuk melihat dan memperbarui parameter kernel Linux-nya:

1. Buka file `/boot/cmdline.txt`. File ini menentukan parameter kernel Linux untuk diterapkan ketika Raspberry Pi boot.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuka file.

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

2. Verifikasi bahwa `/boot/cmdline.txt` file tersebut berisi parameter kernel berikut. `systemd.unified_cgroup_hierarchy=0`Parameter menentukan untuk menggunakan cgroups v1 bukan cgroups v2.

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

   Jika `/boot/cmdline.txt` file tidak berisi parameter ini, atau berisi parameter ini dengan nilai yang berbeda, perbarui file untuk berisi parameter dan nilai ini.

3. Jika Anda memperbarui `/boot/cmdline.txt` file, reboot Raspberry Pi untuk menerapkan perubahan.

   ```
   sudo reboot
   ```

Siapkan perangkat Linux (lainnya)

Untuk mengatur perangkat Linux untuk AWS IoT Greengrass V2

1. Pasang waktu aktif Java, yang diperlukan oleh perangkat lunak inti AWS IoT Greengrass agar berjalan. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan. Perintah berikut menunjukkan cara menginstal OpenJDK di perangkat Anda.

   - Untuk distribusi berbasis Debian atau berbasis Ubuntu:

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

   - Untuk distribusi berbasis Red Hat:

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

   - Untuk Amazon Linux 2:

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

   - Untuk Amazon Linux 2023:

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

   Ketika instalasi selesai, jalankan perintah berikut untuk memverifikasi bahwa Java berjalan pada perangkat Linux Anda.

   ```
   java -version
   ```

   Perintah mencetak versi Java yang berjalan pada perangkat. Misalnya, pada distribusi berbasis Debian, output mungkin terlihat mirip dengan sampel berikut.

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

2. (Opsional) Buat pengguna dan grup sistem default yang menjalankan komponen pada perangkat. Anda juga dapat memilih untuk membiarkan penginstal perangkat lunak AWS

IoT Greengrass Core membuat pengguna dan grup ini selama instalasi dengan argumen `--component-default-user` installer. Untuk informasi selengkapnya, lihat Argumen penginstal.

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

3. Verifikasi bahwa pengguna yang menjalankan perangkat lunak AWS IoT Greengrass Core (biasanya`root`), memiliki izin untuk menjalankan `sudo` dengan pengguna dan grup apa pun.

   a. Jalankan perintah berikut untuk membuka `/etc/sudoers` file.

   ```
   sudo visudo
   ```

   b. Verifikasi bahwa izin untuk pengguna terlihat seperti contoh berikut.

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

4. (Opsional) Untuk menjalankan fungsi Lambda kontainer, Anda harus mengaktifkan cgroups v1, dan Anda harus mengaktifkan dan memasang memori dan perangkat cgroups. Jika Anda tidak berencana untuk menjalankan fungsi Lambda kontainer, Anda dapat melewati langkah ini.

   Untuk mengaktifkan opsi cgroups ini, boot perangkat dengan parameter kernel Linux berikut.

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

   Untuk informasi tentang melihat dan menyetel parameter kernel untuk perangkat Anda, lihat dokumentasi untuk sistem operasi dan boot loader Anda. Ikuti instruksi untuk mengatur parameter kernel secara permanen.

5. Instal semua dependensi lain yang diperlukan pada perangkat Anda seperti yang ditunjukkan oleh daftar persyaratan di. Persyaratan perangkat

Mengatur perangkat Windows

Untuk mengatur perangkat Windows untuk AWS IoT Greengrass V2

1. Pasang waktu aktif Java, yang diperlukan oleh perangkat lunak inti AWS IoT Greengrass agar berjalan. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan.

2. Periksa apakah Java tersedia pada variabel sistem PATH, dan tambahkan jika tidak. LocalSystem Akun menjalankan perangkat lunak AWS IoT Greengrass Core, jadi Anda harus menambahkan Java ke variabel sistem PATH alih-alih variabel pengguna PATH untuk pengguna Anda. Lakukan hal-hal berikut:

   a. Tekan tombol Windows untuk membuka menu mulai.

   b. Ketik **`environment variables`** untuk mencari opsi sistem dari menu mulai.

   c. Di hasil pencarian menu mulai, pilih Edit variabel lingkungan sistem untuk membuka jendela Properti sistem.

   d. Pilih variabel Lingkungan... untuk membuka jendela Variabel Lingkungan.

   e. Di bawah Variabel sistem, pilih Path, lalu pilih Edit. Di jendela variabel Edit lingkungan, Anda dapat melihat setiap jalur pada baris terpisah.

   f. Periksa apakah jalur ke `bin` folder instalasi Java ada. Jalannya mungkin terlihat mirip dengan contoh berikut.

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

   g. Jika `bin` folder instalasi Java hilang dari Path, pilih Baru untuk menambahkannya, lalu pilih OK.

3. Buka Windows Command Prompt (`cmd.exe`) sebagai administrator.

4. Buat pengguna default di LocalSystem akun di perangkat Windows. Ganti *`kata sandi`* dengan kata sandi yang aman.

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

   > **Tip**
   >
   > Bergantung pada konfigurasi Windows Anda, kata sandi pengguna mungkin diatur untuk kedaluwarsa pada tanggal di masa mendatang. Untuk memastikan aplikasi Greengrass Anda terus beroperasi, lacak kapan kata sandi kedaluwarsa, dan perbarui sebelum kedaluwarsa. Anda juga dapat mengatur kata sandi pengguna agar tidak pernah kedaluwarsa.
   >
   > - Untuk memeriksa kapan pengguna dan kata sandinya kedaluwarsa, jalankan perintah berikut.

```
net user ggc_user | findstr /C:expires
```

- Untuk mengatur kata sandi pengguna agar tidak pernah kedaluwarsa, jalankan perintah berikut.

```
wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
```

- Jika Anda menggunakan Windows 10 atau yang lebih baru di mana wmicperintah tidak digunakan lagi, jalankan perintah berikut. PowerShell

```
Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
 'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
```

5. Unduh dan instal PsExecutilitas dari Microsoft pada perangkat.
6. Gunakan PsExec utilitas untuk menyimpan nama pengguna dan kata sandi untuk pengguna default dalam contoh Credential Manager untuk LocalSystem akun tersebut. Ganti *kata sandi* dengan kata sandi pengguna yang Anda tetapkan sebelumnya.

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

Jika PsExec License Agreementterbuka, pilih Acceptuntuk menyetujui lisensi dan jalankan perintah.

> **Note**
>
> Pada perangkat Windows, LocalSystem akun menjalankan inti Greengrass, dan Anda harus menggunakan utilitas untuk menyimpan PsExec informasi pengguna default di akun. LocalSystem Menggunakan aplikasi Credential Manager menyimpan informasi ini di akun Windows dari pengguna yang saat ini masuk, bukan LocalSystem akun.

Langkah 3: Instal perangkat lunakAWS IoT Greengrass inti

Ikuti langkah-langkah di bagian ini untuk mengatur Raspberry Pi Anda sebagai perangkat inti AWS IoT Greengrass yang dapat Anda gunakan untuk pengembangan lokal. Dalam bagian ini, Anda men-download dan menjalankan installer yang melakukan hal-hal berikut untuk mengonfigurasi perangkat lunak inti AWS IoT Greengrass untuk perangkat Anda:

- Menginstal komponen inti Greengrass. Nucleus adalah komponen wajib dan persyaratan minimum untuk menjalankan perangkat lunakAWS IoT Greengrass inti pada perangkat. Untuk informasi selengkapnya, lihat, komponen nukleus Greengrass.
- Daftarkan perangkat Anda sebagai AWS IoT dan unduh sertifikat digital yang memungkinkan perangkat Anda terhubung ke AWS. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.
- Menambahkan objek AWS IoT perangkat ke grup objek, yang merupakan grup atau armada objek AWS IoT. Grup objek memungkinkan Anda untuk mengelola armada perangkat inti Greengrass. Saat men-deploy komponen perangkat lunak ke perangkat, Anda dapat memilih untuk men-deploy ke perangkat individual atau ke grup perangkat. Untuk informasi selengkapnya, lihat, Mengelola perangkat dengan AWS IoT di Panduan Developer AWS IoT Core.
- Buat IAM role yang memungkinkan perangkat inti Greengrass Anda untuk berinteraksi dengan layanan AWS. Secara default, peran ini memungkinkan perangkat Anda berinteraksi denganAWS IoT dan mengirim log ke AmazonCloudWatch Logs. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.
- Menginstal antarmuka baris perintah AWS IoT Greengrass (`greengrass-cli`), yang dapat Anda gunakan untuk menguji komponen kustom yang Anda kembangkan pada perangkat inti. Untuk informasi selengkapnya, lihat Antarmuka Baris Perintah Greengrass.

Instal perangkat lunak AWS IoT Greengrass Core (konsol)

1. Masuk ke konsol AWS IoT Greengrass tersebut.
2. Di bawah Memulai Greengrass, pilih Siapkan satu perangkat inti.
3. Di bawah Langkah 1: Daftarkan perangkat inti Greengrass, untuk nama perangkat Core, masukkan nama AWS IoT benda untuk perangkat inti Greengrass Anda. Jika objek tidak ada, installer akan membuatnya.
4. Di bawah Langkah 2: Tambahkan ke grup hal untuk menerapkan penerapan berkelanjutan, untuk grup Thing, pilih grup AWS IoT hal yang ingin Anda tambahkan perangkat inti Anda.

    - Jika Anda memilih Masukkan nama grup baru, lalu di Nama grup Thing, masukkan nama grup baru yang akan dibuat. Installer membuat grup baru untuk Anda.
    - Jika Anda memilih Pilih grup yang ada, lalu di Nama grup Thing, pilih grup yang ada yang ingin Anda gunakan.
    - Jika Anda memilih Tidak ada grup, maka penginstal tidak menambahkan perangkat inti ke grup sesuatu.

5. Di bawah Langkah 3: Instal perangkat lunak Greengrass Core, selesaikan langkah-langkah berikut.

   a. Pilih sistem operasi perangkat inti Anda: Linux atau Windows.

   b. Berikan AWS kredensi Anda ke perangkat sehingga penginstal dapat menyediakan sumber daya AWS IoT dan IAM untuk perangkat inti Anda. Untuk meningkatkan keamanan, sebaiknya Anda mendapatkan kredensi sementara untuk peran IAM yang hanya mengizinkan izin minimum yang diperlukan untuk penyediaan. Untuk informasi selengkapnya, lihat Kebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya.

      > **Note**
      >
      > Penginstal tidak menyimpan atau menyimpan kredensial Anda.

      Di perangkat Anda, lakukan salah satu hal berikut untuk mengambil kredensi dan membuatnya tersedia untuk penginstal perangkat lunak AWS IoT Greengrass Core:

      - (Disarankan) Gunakan kredenal temporay dari AWS IAM Identity Center

        i. Berikan ID kunci akses, kunci akses rahasia, dan token sesi dari Pusat Identitas IAM. Untuk informasi selengkapnya, lihat Penyegaran kredenal manual di Mendapatkan dan menyegarkan kredenal sementara di panduan pengguna Pusat Identitas IAM.

        ii. Jalankan perintah berikut untuk memberikan kredensial ke perangkat lunak inti AWS IoT Greengrass.

        Linux or Unix

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

        Windows Command Prompt (CMD)

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

```
set AWS_SESSION_TOKEN=AQoDYXdzEJr1K...o5OytwEXAMPLE=
```

PowerShell

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

- Gunakan kredenal keamanan sementara dari peran IAM:

  i. Berikan access key ID, secret access key, dan token sesi dari IAM role yang Anda teruskan. Untuk informasi selengkapnya tentang cara mengambil kredenal ini, lihat Meminta kredenal keamanan sementara di Panduan Pengguna IAM.

  ii. Jalankan perintah berikut untuk memberikan kredensial ke perangkat lunak inti AWS IoT Greengrass.

  Linux or Unix

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

  Windows Command Prompt (CMD)

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

  PowerShell

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

- Gunakan kredensial jangka panjang dari pengguna IAM:

  i. Berikan access key ID dan secret access key untuk pengguna IAM Anda. Anda dapat membuat pengguna IAM untuk penyediaan yang kemudian Anda hapus.

Untuk kebijakan IAM untuk memberikan pengguna, lihatKebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya. Untuk informasi selengkapnya tentang cara mengambil kredensial jangka panjang, lihat Mengelola access key untuk pengguna IAM di Panduan Pengguna IAM.

ii. Jalankan perintah berikut untuk memberikan kredensial ke perangkat lunak inti AWS IoT Greengrass.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

iii. (Opsional) Jika Anda membuat pengguna IAM untuk menyediakan perangkat Greengrass Anda, hapus pengguna tersebut.

iv. (Opsional) Jika Anda menggunakan ID kunci akses dan kunci akses rahasia dari pengguna IAM yang ada, perbarui kunci untuk pengguna sehingga tidak lagi valid. Untuk informasi selengkapnya, lihat Memperbarui kunci akses di panduan AWS Identity and Access Management pengguna.

c. Di bawah Jalankan penginstal, selesaikan langkah-langkah berikut.

i. Di bawah Unduh penginstal, pilih Salin dan jalankan perintah yang disalin pada perangkat inti Anda. Perintah ini mengunduh versi terbaru dari perangkat lunak AWS IoT Greengrass Core dan membuka ritsletingnya di perangkat Anda.

ii. Di bawah Jalankan penginstal, pilih Salin, dan jalankan perintah yang disalin pada perangkat inti Anda. Perintah ini menggunakan nama grup AWS IoT benda dan benda

yang Anda tentukan sebelumnya untuk menjalankan penginstal perangkat lunak AWS IoT Greengrass Core dan menyiapkan AWS sumber daya untuk perangkat inti Anda.

Perintah ini juga melakukan hal berikut:

- Siapkan perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem yang berjalan saat boot. Pada perangkat Linux, ini membutuhkan sistem init Systemd.

  > **Important**
  >
  > Pada perangkat inti Windows, Anda harus mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

- Terapkan komponen AWS IoT GreengrassCLI, yang merupakan alat baris perintah yang memungkinkan Anda mengembangkan komponen Greengrass khusus pada perangkat inti.
- Tentukan untuk menggunakan pengguna `ggc_user` sistem untuk menjalankan komponen perangkat lunak pada perangkat inti. Pada perangkat Linux, perintah ini juga menentukan untuk menggunakan grup `ggc_group` sistem, dan penginstal membuat pengguna dan grup sistem untuk Anda.

Ketika Anda menjalankan perintah ini, Anda akan melihat pesan berikut untuk menunjukkan bahwa installer berhasil.

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

> **Note**
>
> Jika Anda memiliki perangkat Linux dan tidak memiliki systemd, penginstal tidak akan mengatur perangkat lunak sebagai layanan sistem, dan Anda tidak akan melihat pesan sukses untuk menyiapkan inti sebagai layanan sistem.

Instal perangkat lunak AWS IoT Greengrass inti (CLI)

Untuk menginstal dan mengonfigurasi perangkat lunak inti AWS IoT Greengrass

1. Pada perangkat inti Greengrass Anda, jalankan perintah berikut untuk beralih ke direktori home.

   Linux or Unix

   ```
   cd ~
   ```

   Windows Command Prompt (CMD)

   ```
   cd %USERPROFILE%
   ```

   PowerShell

   ```
   cd ~
   ```

2. Di perangkat inti Anda, unduh perangkat lunak AWS IoT Greengrass Core ke file bernama`greengrass-nucleus-latest.zip`.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

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

3. Unzip perangkat lunak inti AWS IoT Greengrass ke folder di perangkat Anda. Ganti *GreengrassInstaller*dengan folder yang ingin Anda gunakan.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

4. Jalankan perintah berikut untuk meluncurkan penginstal perangkat lunak inti AWS IoT Greengrass. Perintah ini melakukan hal berikut:

   - Buat sumber daya AWS yang dibutuhkan perangkat inti untuk beroperasi.
   - Siapkan perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem yang berjalan saat boot. Pada perangkat Linux, ini membutuhkan sistem init Systemd.

     > **Important**
     >
     > Pada perangkat inti Windows, Anda harus mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

   - Terapkan komponen AWS IoT GreengrassCLI, yang merupakan alat baris perintah yang memungkinkan Anda mengembangkan komponen Greengrass khusus pada perangkat inti.
   - Tentukan untuk menggunakan pengguna `ggc_user` sistem untuk menjalankan komponen perangkat lunak pada perangkat inti. Pada perangkat Linux, perintah ini juga menentukan untuk menggunakan grup `ggc_group` sistem, dan penginstal membuat pengguna dan grup sistem untuk Anda.

Ganti nilai argumen dalam perintah Anda sebagai berikut.

a. */greengrass/v2*atau *C:\greengrass\v2*: Jalur ke folder root yang akan digunakan untuk menginstal perangkat lunak AWS IoT Greengrass Core.

b. *GreengrassInstaller*. Path ke folder tempat Anda membongkar penginstal perangkat lunak inti AWS IoT Greengrass.

c. *region*. Wilayah AWS tempat untuk menemukan atau membuat sumber daya.

d. *MyGreengrassCore*. Nama objek AWS IoT untuk perangkat inti Greengrass Anda. Jika objek tidak ada, installer akan membuatnya. Penginstal mengunduh sertifikat tersebut untuk diautentikasi sebagai objek AWS IoT. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.

> **Note**
>
> Nama objek tidak dapat berisi karakter titik dua (`:`).

e. *MyGreengrassCoreGroup*. Nama grup objek AWS IoT untuk perangkat inti Greengrass Anda. Jika grup objek tidak ada, installer akan membuatnya dan menambahkan objek padanya. Jika grup objek ada dan memiliki deployment yang aktif, perangkat inti akan men-download dan menjalankan perangkat lunak yang ditetapkan oleh deployment.

> **Note**
>
> Nama grup objek tidak dapat berisi karakter titik dua (`:`).

f. *ThingPolicyGreenGrassV2IoT*. Nama AWS IoT kebijakan yang memungkinkan perangkat inti Greengrass untuk berkomunikasi dengan dan. AWS IoT AWS IoT Greengrass Jika AWS IoT kebijakan tidak ada, penginstal akan membuat AWS IoT kebijakan permisif dengan nama ini. Anda dapat membatasi izin kebijakan ini untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2.

g. *GreenGrassV2 TokenExchangeRole*. Nama IAM role yang memungkinkan perangkat inti Greengrass untuk mendapatkan kredensial AWS sementara. Jika peran itu tidak ada, penginstal akan membuatnya dan membuat serta melampirkan kebijakan bernama *GreengrassV2TokenExchangeRole*`Access`. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

h. *GreengrassCoreTokenExchangeRoleAlias*. Nama IAM role yang memungkinkan perangkat inti Greengrass untuk mendapatkan kredensial sementara nanti. Jika alias peran tidak ada, penginstal akan membuatnya dan mengarahkannya ke IAM role yang Anda tentukan. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

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

> **Note**
>
> Jika Anda menjalankan AWS IoT Greengrass pada perangkat dengan memori terbatas, Anda dapat mengendalikan jumlah memori yang digunakan oleh perangkat lunak inti AWS IoT Greengrass. Untuk mengontrol alokasi memori, Anda dapat mengatur pilihan ukuran tumpukan JVM di konfigurasi parameter `jvm0ptions` dalam komponen nukleus anda. Untuk informasi selengkapnya, lihat Kontrol alokasi memori dengan opsi JVM.

Ketika Anda menjalankan perintah ini, Anda akan melihat pesan berikut untuk menunjukkan bahwa installer berhasil.

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

> **Note**
>
> Jika Anda memiliki perangkat Linux dan tidak memiliki systemd, penginstal tidak akan mengatur perangkat lunak sebagai layanan sistem, dan Anda tidak akan melihat pesan sukses untuk menyiapkan inti sebagai layanan sistem.

(Opsional) Jalankan perangkat lunak Greengrass (Linux)

Jika Anda menginstal perangkat lunak sebagai layanan sistem, installer akan menjalankan perangkat lunak untuk Anda. Jika tidak, Anda harus menjalankan perangkat lunak itu. Untuk melihat apakah installer mengatur perangkat lunak sebagai layanan sistem, cari baris berikut dalam output installer.

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

Jika Anda tidak melihat pesan ini, lakukan langkah-langkah berikut untuk menjalankan perangkat lunak:

1. Jalankan perintah berikut untuk menjalankan perangkat lunak tersebut.

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

   Perangkat lunak mencetak pesan berikut jika berhasil meluncurkan.

   ```
   Launched Nucleus successfully.
   ```

2. Anda harus membiarkan shell perintah saat ini terbuka untuk menjaga perangkat lunak AWS IoT Greengrass Core tetap berjalan. Jika Anda menggunakan SSH untuk terhubung ke perangkat inti, jalankan perintah berikut di komputer pengembangan Anda untuk membuka sesi SSH kedua yang dapat Anda gunakan untuk menjalankan perintah tambahan pada perangkat inti. Ganti *nama* pengguna dengan nama pengguna untuk masuk, dan ganti *pi-ip-address*dengan alamat IP perangkat.

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

Untuk informasi lebih lanjut tentang cara berinteraksi dengan layanan sistem Greengrass, lihat Konfigurasikan inti Greengrass sebagai layanan sistem.

Verifikasi instalasi CLI Greengrass di perangkat

CLI Greengrass dapat memakan waktu hingga satu menit untuk digunakan. Jalankan perintah berikut untuk memeriksa status penyebaran. Ganti *MyGreengrassCore* dengan nama perangkat inti Anda.

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

`coreDeviceExecutionStatus` menunjukkan status deployment ke perangkat inti. Ketika statusnya adalah `SUCCEEDED`, jalankan perintah berikut untuk memverifikasi bahwa Greengrass CLI terinstal dan berjalan. Ganti */greengrass/v2* dengan jalur ke folder root.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Perintah output membantu informasi untuk Greengrass CLI. Jika `greengrass-cli` tidak ditemukan, deployment mungkin gagal untuk menginstal Greengrass CLI. Untuk informasi selengkapnya, lihat Pemecahan masalah AWS IoT Greengrass V2.

Anda juga dapat menjalankan perintah berikut untuk menyebarkan AWS IoT Greengrass CLI secara manual ke perangkat Anda.

- Ganti *wilayah* dengan Wilayah AWS yang Anda gunakan. Pastikan Anda menggunakan yang sama dengan Wilayah AWS yang Anda gunakan untuk mengonfigurasi AWS CLI pada perangkat Anda.
- Ganti *account-id dengan ID* Anda Akun AWS .
- Ganti *MyGreengrassCore* dengan nama perangkat inti Anda.

Linux, macOS, or Unix

```
aws greengrassv2 create-deployment \
  --target-arn "arn:aws:iot:region:account-id:thing/MyGreengrassCore" \
  --components '{
    "aws.greengrass.Cli": {
      "componentVersion": "2.12.6"
    }
```

```
  }'
```

Windows Command Prompt (CMD)

```
aws greengrassv2 create-deployment ^
  --target-arn "arn:aws:iot:region:account-id:thing/MyGreengrassCore" ^
  --components "{\"aws.greengrass.Cli\":{\"componentVersion\":\"2.12.6\"}}"
```

PowerShell

```
aws greengrassv2 create-deployment `
  --target-arn "arn:aws:iot:region:account-id:thing/MyGreengrassCore" `
  --components '{\"aws.greengrass.Cli\":{\"componentVersion\":\"2.12.6\"}}'
```

> **Tip**
>
> Anda dapat menambahkan */greengrass/v2*/bin (Linux) atau *C:\greengrass\v2*\bin (Windows) ke variabel PATH lingkungan Anda untuk berjalan `greengrass-cli` tanpa jalur absolutnya.

Perangkat lunak inti AWS IoT Greengrass dan alat pengembangan lokal berjalan di perangkat Anda. Selanjutnya, Anda dapat mengembangkan komponen Hello World AWS IoT Greengrass di perangkat Anda.

Langkah 4: Kembangkan dan uji komponen di perangkat Anda

Komponen adalah modul perangkat lunak yang berjalan pada perangkat AWS IoT Greengrass inti. Komponen memungkinkan Anda untuk membuat dan mengelola aplikasi yang kompleks sebagai blok bangunan diskrit yang dapat Anda gunakan kembali dari satu perangkat inti Greengrass ke yang lain. Setiap komponen terdiri atas resep dan artifact.

- Resep

  Setiap komponen berisi file resep, yang mendefinisikan metadata nya. Resep ini juga menentukan parameter konfigurasi komponen, dependensi komponen, siklus hidup, dan kompatibilitas platform. Siklus hidup komponen menentukan perintah yang menginstal, menjalankan, dan menutup komponen. Untuk informasi selengkapnya, lihat AWS IoT Greengrass referensi resep komponen.

Anda bisa menentukan resep dalam format JSON atau YAML.

- Artefak

  Komponen dapat memiliki sejumlah artifact, yang merupakan komponen biner. Artifact dapat mencakup skrip, kode yang dikompilasi, sumber daya statis, dan file lain yang dikonsumsi komponen. Komponen juga dapat mengonsumsi artifact dari dependensi komponen.

Dengan AWS IoT Greengrass, Anda dapat menggunakan CLI Greengrass untuk mengembangkan dan menguji komponen secara lokal pada perangkat inti Greengrass tanpa interaksi dengan Cloud. AWS Saat menyelesaikan komponen lokal, Anda dapat menggunakan resep komponen dan artefak untuk membuat komponen tersebut di AWS IoT Greengrass layanan di AWS Cloud, lalu menerapkannya ke semua perangkat inti Greengrass Anda. Untuk informasi selengkapnya tentang komponen, lihatKembangkan AWS IoT Greengrass komponen.

Di bagian ini, Anda mempelajari cara membuat dan menjalankan komponen Hello World dasar secara lokal di perangkat inti Anda.

Untuk mengembangkan komponen Hello World di perangkat Anda

1. Buat folder untuk komponen Anda dengan subfolder untuk resep dan artefak. Jalankan perintah berikut pada perangkat inti Greengrass Anda untuk membuat folder ini dan mengubah ke folder komponen. Ganti *~/greengrassv2* atau *%USERPROFILE%\ greengrassv2* dengan path ke folder yang akan digunakan untuk pengembangan lokal.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

2. Gunakan editor teks untuk membuat file resep yang mendefinisikan metadata, parameter, dependensi, siklus hidup, dan kemampuan platform komponen Anda. Sertakan versi komponen dalam nama file resep agar Anda dapat mengidentifikasi resep yang mencerminkan versi komponen. Anda dapat memilih format YAML atau JSON untuk resep Anda.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   JSON

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

   YAML

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

   > **Note**
   >
   > AWS IoT Greengrass menggunakan versi semantik untuk komponen. Versi semantik mengikuti sistem nomor mayor.minor.patch. Sebagai contoh, versi `1.0.0` merupakan rilis mayor pertama untuk sebuah komponen. Untuk informasi lebih lanjut, lihat spesifikasi versi semantik.

3. Tempel resep berikut ke file.

   JSON

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

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

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.HelloWorld
ComponentVersion: '1.0.0'
ComponentDescription: My first AWS IoT Greengrass component.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    Message: world
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      run: |
        python3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
  - Platform:
      os: windows
    Lifecycle:
      run: |
        py -3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
```

Bagian `ComponentConfiguration` resep ini menentukan parameter, `Message`, yang default -nya adalah `world`. Bagian `Manifests` menentukan manifes, yang merupakan serangkaian instruksi siklus hidup dan artifact untuk platform. Anda dapat menentukan beberapa manifes untuk menentukan petunjuk pemasangan yang berbeda untuk berbagai platform, misalnya. Dalam manifes, bagian `Lifecycle` memerintahkan perangkat inti Greengrass untuk menjalankan skrip Hello World dengan nilai parameter `Message` sebagai argumen.

4. Jalankan perintah berikut untuk membuat folder untuk komponen artifact.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

   Important

   Anda harus menggunakan format berikut untuk jalur folder artifact. Sertakan nama dan versi komponen yang Anda tentukan dalam resep.

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

5. Gunakan editor teks untuk membuat file artefak skrip Python untuk komponen Hello World Anda.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

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

   Copy dan paste script Python berikut ke dalam file.

```
import sys

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

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

6. Gunakan AWS IoT Greengrass CLI lokal untuk mengelola komponen pada perangkat inti Greengrass Anda.

   Jalankan perintah berikut untuk menyebarkan komponen ke AWS IoT Greengrass inti. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan folder AWS IoT Greengrass V2 root Anda, dan ganti *~/greengrassv2 atau% USERPROFILE%\ greengrassv2* dengan folder pengembangan komponen Anda.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

   Perintah ini menambahkan komponen yang menggunakan resep di `recipes` dan skrip Python di `artifacts`. Opsi `--merge` menambahkan atau memperbarui komponen dan versi yang Anda tentukan.

7. Perangkat lunak AWS IoT Greengrass Core menyimpan stdout dari proses komponen ke file log di folder. `logs` Jalankan perintah berikut untuk memverifikasi bahwa komponen Hello World berjalan dan mencetak pesan.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   `type`Perintah menulis konten file ke terminal. Jalankan perintah ini beberapa kali untuk mengamati perubahan dalam file.

   PowerShell

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

   Anda akan melihat pesan yang mirip dengan contoh berikut ini.

   ```
   Hello, world!
   ```

   > **Note**
   >
   > Jika file tidak ada, penyebaran lokal mungkin belum lengkap. Jika file tidak ada dalam waktu 15 detik, deployment mungkin gagal. Hal ini dapat terjadi jika resep Anda tidak valid, misalnya. Jalankan perintah berikut untuk melihat file log AWS IoT Greengrass inti. File ini mencakup log dari layanan deployment perangkat inti Greengrass.
   >
   > Linux or Unix
   >
   > ```
   > sudo tail -f /greengrass/v2/logs/greengrass.log
   > ```
   >
   > Windows Command Prompt (CMD)
   >
   > ```
   > type C:\greengrass\v2\logs\greengrass.log
   > ```

`type`Perintah menulis konten file ke terminal. Jalankan perintah ini beberapa kali untuk mengamati perubahan dalam file.

PowerShell

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

8. Ubah komponen lokal untuk mengiterasi dan menguji kode Anda. Buka `hello_world.py` di editor teks, dan tambahkan kode berikut pada baris 4 untuk mengedit pesan yang dicatat oleh AWS IoT Greengrass inti.

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

Skrip `hello_world.py` sekarang akan berisi konten berikut.

```
import sys

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

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

9. Jalankan perintah berikut untuk memperbarui komponen dengan perubahan Anda.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Perintah ini memperbarui `com.example.HelloWorld` komponen dengan artefak Hello World terbaru.

10. Jalankan perintah berikut untuk me-restart komponen. Ketika Anda me-restart komponen, perangkat inti menggunakan perubahan terakhir.

    Linux or Unix

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

    Windows Command Prompt (CMD)

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

    PowerShell

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

11. Periksa log lagi untuk memverifikasi bahwa komponen Hello World mencetak pesan baru.

    Linux or Unix

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

    Windows Command Prompt (CMD)

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

`type`Perintah menulis konten file ke terminal. Jalankan perintah ini beberapa kali untuk mengamati perubahan dalam file.

PowerShell

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

Anda akan melihat pesan yang mirip dengan contoh berikut ini.

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

12. Anda dapat memperbarui parameter konfigurasi komponen untuk menguji konfigurasi yang berbeda. Ketika Anda men-deploy komponen, Anda dapat menentukan pembaruan konfigurasi, yang menentukan cara mengubah konfigurasi komponen pada perangkat inti. Anda dapat menentukan nilai konfigurasi yang akan di-reset ke nilai default dan nilai-nilai konfigurasi baru yang akan digabungkan ke perangkat inti. Untuk informasi selengkapnya, lihat Perbarui konfigurasi komponen.

    Lakukan hal-hal berikut:

    a. Gunakan editor teks untuk membuat file yang dipanggil `hello-world-config-update.json` untuk memuat pembaruan konfigurasi

       Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

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

    b. Salin dan tempel objek JSON berikut ke dalam file. Objek JSON ini menentukan pembaruan konfigurasi yang menggabungkan nilai `friend` ke parameter `Message` untuk memperbarui nilainya. Pembaruan konfigurasi ini tidak menentukan nilai apa pun yang akan di-reset. Anda tidak perlu mengatur ulang parameter `Message` karena pembaruan merge menggantikan nilai yang ada.

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

```
  }
}
```

c. Jalankan perintah berikut untuk men-deploy pembaruan konfigurasi pada komponen Hello World.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

d. Periksa log lagi untuk memverifikasi bahwa komponen Hello World mengeluarkan pesan baru.

Linux or Unix

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

Windows Command Prompt (CMD)

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

`type`Perintah menulis konten file ke terminal. Jalankan perintah ini beberapa kali untuk mengamati perubahan dalam file.

PowerShell

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

Anda akan melihat pesan yang mirip dengan contoh berikut ini.

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

13. Setelah Anda selesai menguji komponen, lepaskan komponen tersebut dari perangkat inti Anda. Jalankan perintah berikut.

    Linux or Unix

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

    Windows Command Prompt (CMD)

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

    PowerShell

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

    > **Important**
    >
    > Langkah ini diperlukan bagi Anda untuk men-deploy komponen kembali ke perangkat inti setelah Anda mengunggahnya ke AWS IoT Greengrass. Jika tidak, deployment akan gagal dengan kesalahan kompatibilitas versi karena deployment lokal menentukan versi yang berbeda dari komponen.

    Jalankan perintah berikut dan verifikasi bahwa perintah `com.example.HelloWorld` tidak muncul dalam daftar komponen di perangkat Anda.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Komponen Hello World Anda selesai, dan sekarang Anda dapat mengunggahnya ke layanan AWS IoT Greengrass cloud. Kemudian, Anda dapat menerapkan komponen ke perangkat inti Greengrass.

Langkah 5: Buat komponen Anda dalam AWS IoT Greengrass layanan

Ketika Anda selesai mengembangkan komponen pada perangkat inti Anda, Anda dapat mengunggahnya ke AWS IoT Greengrass layanan di AWS Cloud. Anda juga dapat langsung membuat komponen di AWS IoT Greengrass konsol. AWS IoT Greengrass menyediakan layanan manajemen komponen yang menghosting komponen Anda sehingga Anda dapat menyebarkannya ke perangkat individual atau armada perangkat. Untuk mengunggah komponen ke AWS IoT Greengrass layanan, Anda menyelesaikan langkah-langkah berikut:

- Unggah artefak komponen ke bucket S3.
- Tambahkan setiap URI Amazon Simple Storage Service (Amazon S3) artefak ke resep komponen.
- Buat komponen AWS IoT Greengrass dari resep komponen.

Di bagian ini, Anda menyelesaikan langkah-langkah ini pada perangkat inti Greengrass Anda untuk mengunggah komponen Hello World Anda ke layanan. AWS IoT Greengrass

Buat komponen Anda di AWS IoT Greengrass (konsol)

1. Gunakan bucket S3 di AWS akun Anda untuk meng-host artefak AWS IoT Greengrass komponen. Saat Anda men-deploy komponen ke perangkat inti, perangkat akan mengunduh artefak komponen dari bucket.

   Anda dapat menggunakan bucket S3 yang sudah ada, atau Anda dapat membuat bucket baru.

   a. Di konsol Amazon S3, di bawah Bucket, pilih Buat ember.

   b. Untuk nama Bucket, masukkan nama bucket yang unik. Misalnya, Anda dapat menggunakan**greengrass-component-artifacts-*region*-*123456789012***. Ganti *123456789012* dengan ID AWS akun dan *wilayah* Anda dengan Wilayah AWS yang Anda gunakan untuk tutorial ini.

   c. Untuk AWS wilayah, pilih AWS Wilayah yang Anda gunakan untuk tutorial ini.

   d. Pilih Buat bucket.

   e. Di bawah Bucket, pilih bucket yang Anda buat, unggah `hello_world.py` skrip ke `artifacts/com.example.HelloWorld/1.0.0` folder di bucket. Untuk informasi tentang mengunggah objek ke bucket S3, lihat Mengunggah objek di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

   f. Salin URI S3 `hello_world.py` objek di bucket S3. URI ini akan terlihat serupa dengan contoh berikut. Ganti DOC-EXAMPLE-BUCKET dengan nama bucket S3.

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

2. Izinkan perangkat inti mengakses artefak komponen dalam bucket S3.

   Setiap perangkat inti memiliki peran IAM perangkat inti yang memungkinkannya berinteraksi dengan AWS IoT dan mengirim log ke AWS Cloud. Peran perangkat ini tidak mengizinkan akses ke bucket S3 secara default, sehingga Anda harus membuat dan melampirkan kebijakan yang memungkinkan perangkat inti mengambil artefak komponen dari bucket S3.

   Jika peran perangkat Anda sudah memungkinkan akses ke bucket S3, Anda dapat melewati langkah ini. Jika tidak, buat kebijakan IAM yang memungkinkan akses dan melampirkannya pada peran, sebagai berikut:

   a. Di menu navigasi konsol IAM, pilih Kebijakan, lalu pilih Buat kebijakan.

b. Pada tab JSON, ganti placeholder konten dengan kebijakan berikut. Ganti DOC-EXAMPLE-BUCKET dengan nama bucket S3 yang berisi artefak komponen untuk diunduh perangkat inti.

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

c. Pilih Selanjutnya.

d. Di bagian Detail kebijakan, untuk Nama, masukkan**`MyGreengrassV2ComponentArtifactPolicy`**.

e. Pilih Buat kebijakan.

f. Di menu navigasi konsol IAM, pilih Peran, lalu pilih nama peran untuk perangkat inti. Anda menentukan nama peran ini saat menginstal perangkat lunak AWS IoT Greengrass Inti. Jika Anda tidak menentukan nama, defaultnya adalah`GreengrassV2TokenExchangeRole`.

g. Di bawah Izin, pilih Tambahkan izin, lalu pilih Lampirkan kebijakan.

h. Pada halaman Tambahkan izin, pilih kotak centang di samping `MyGreengrassV2ComponentArtifactPolicy` kebijakan yang Anda buat, lalu pilih Tambahkan izin.

3. Gunakan resep komponen untuk membuat komponen di AWS IoT Greengrass konsol.

a. Di menu navigasi AWS IoT Greengrass konsol, pilih Komponen, lalu pilih Buat komponen.

b. Di bawah Informasi komponen, pilih Masukkan resep sebagai JSON. Resep placeholder akan terlihat mirip dengan contoh berikut.

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.HelloWorld",
  "ComponentVersion": "1.0.0",
```

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

c. Ganti URI placeholder di setiap `Artifacts` bagian dengan URI S3 objek Anda. `hello_world.py`

d. Pilih Buat komponen.

e. Di com.example. HelloWorldhalaman komponen, verifikasi bahwa Status komponen adalah Deployable.

Buat komponen Anda di AWS IoT Greengrass (AWS CLI)

Untuk mengunggah komponen Hello World anda

1. Gunakan bucket S3 di artefak AWS IoT Greengrass komponen Anda Akun AWS untuk meng-host. Saat Anda men-deploy komponen ke perangkat inti, perangkat akan mengunduh artefak komponen dari bucket.

   Anda dapat menggunakan bucket S3 yang ada, atau menjalankan perintah berikut untuk membuat bucket. Perintah ini membuat bucket dengan Akun AWS ID Anda dan Wilayah AWS untuk membentuk nama bucket yang unik. Ganti *123456789012* dengan Akun AWS ID dan *wilayah* Anda dengan Wilayah AWS yang Anda gunakan untuk tutorial ini.

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

   Perintah ini mengeluarkan informasi berikut jika permintaan berhasil.

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

2. Izinkan perangkat inti mengakses artefak komponen dalam bucket S3.

   Setiap perangkat inti memiliki peran IAM perangkat inti yang memungkinkannya berinteraksi dengan AWS IoT dan mengirim log ke file. AWS Cloud Peran perangkat ini tidak mengizinkan akses ke bucket S3 secara default, sehingga Anda harus membuat dan melampirkan kebijakan yang memungkinkan perangkat inti mengambil artefak komponen dari bucket S3.

   Jika peran perangkat inti sudah memungkinkan akses ke bucket S3, Anda dapat melewati langkah ini. Jika tidak, buat kebijakan IAM yang memungkinkan akses dan melampirkannya pada peran, sebagai berikut:

   a. Buat file bernama `component-artifact-policy.json` dan salin JSON berikut ke dalam file. Kebijakan ini memungkinkan akses ke semua file dalam bucket S3. Ganti DOC-EXAMPLE-BUCKET dengan nama bucket S3.

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

b. Jalankan perintah berikut untuk membuat kebijakan dari dokumen kebijakan di `component-artifact-policy.json`.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Salin Amazon Resource Name (ARN) kebijakan dari metadata kebijakan dalam output. Anda menggunakan ARN ini untuk melampirkan kebijakan ini ke peran perangkat inti di langkah berikutnya.

c. Jalankan perintah berikut untuk melampirkan kebijakan tersebut pada peran perangkat inti. Ganti *GreenGrassV2 TokenExchangeRole* dengan nama peran untuk perangkat inti.

Anda menentukan nama peran ini saat menginstal perangkat lunak AWS IoT Greengrass Inti. Ganti ARN kebijakan dengan ARN dari langkah sebelumnya.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Jika perintah tidak memiliki output, itu berhasil. Perangkat inti sekarang dapat mengakses artefak yang Anda unggah ke bucket S3 ini.

3. Unggah artefak skrip Python Hello World ke bucket S3.

Jalankan perintah berikut untuk mengunggah skrip ke jalur yang sama di bucket tempat skrip ada di AWS IoT Greengrass inti Anda. Ganti DOC-EXAMPLE-BUCKET dengan nama bucket S3.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Perintah itu mengeluarkan baris yang dimulai dengan `upload:` jika permintaan berhasil.

4. Tambahkan URI Amazon S3 artefak ke resep komponen.

   URI Amazon S3 terdiri atas nama bucket dan path ke objek artefak dalam bucket. URI Amazon S3 Anda adalah URI tujuan Anda mengunggah artefak pada langkah sebelumnya. URI ini akan terlihat serupa dengan contoh berikut. Ganti DOC-EXAMPLE-BUCKET dengan nama bucket S3.

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

   Untuk menambahkan artefak ke resep, tambahkan daftar `Artifacts` yang berisi struktur dengan URI Amazon S3.

   JSON

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

   Buka file resep di editor teks.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

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

Tambahkan artefak ke resep. File resep Anda akan terlihat seperti contoh berikut.

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

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

YAML

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

Buka file resep di editor teks.

Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

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

Tambahkan artefak ke resep. File resep Anda akan terlihat seperti contoh berikut.

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

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

5. Buat sumber daya komponen AWS IoT Greengrass dari resep. Jalankan perintah berikut untuk membuat komponen dari resep, yang Anda sediakan sebagai file biner.

   JSON

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

   YAML

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

   Responsnya terlihat seperti contoh berikut jika permintaan berhasil.

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

   Salin `arn` dari output untuk memeriksa keadaan komponen pada langkah berikutnya.

> **Note**
>
> Anda juga dapat melihat komponen Hello World di Konsol AWS IoT Greengrass pada halaman Komponen.

6. Verifikasi bahwa komponen tersebut terbuat dan siap untuk di-deploy. Ketika Anda membuat komponen, keadaannya adalah `REQUESTED`. Kemudian, AWS IoT Greengrass memvalidasi bahwa komponen tersebut dapat di-deploy. Anda dapat menjalankan perintah berikut untuk melakukan kueri atas status komponen dan memverifikasi bahwa komponen Anda dapat di-deploy. Ganti `arn` dengan ARN dari langkah sebelumnya.

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

Jika komponen tervalidasi, respons akan menunjukkan bahwa keadaan komponen adalah `DEPLOYABLE`.

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

Komponen Hello World Anda sekarang tersedia di AWS IoT Greengrass. Anda dapat men-deploy-nya kembali ke perangkat inti Greengrass ini atau ke perangkat inti lainnya.

Langkah 6: Terapkan komponen Anda

Dengan AWS IoT Greengrass, Anda dapat men-deploy komponen ke perangkat individual atau grup perangkat. Saat Anda men-deploy komponen, AWS IoT Greengrass akan menginstal dan menjalankan perangkat lunak komponen tersebut pada setiap perangkat target. Anda menentukan komponen mana yang akan di-deploy dan pembaruan konfigurasi yang akan di-deploy pada setiap komponen. Anda juga dapat mengontrol bagaimana deployment mulai tersedia pada perangkat yang menjadi target deployment. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat.

Di bagian ini, Anda menerapkan komponen Hello World Anda kembali ke perangkat inti Greengrass Anda.

Terapkan komponen Anda (konsol)

1. Pada menu navigasi konsol AWS IoT Greengrass tersebut, pilih Komponen.
2. Pada halaman Components, pada tab My components, pilih com.example.HelloWorld.
3. Pada halaman com.example.HelloWorld pilih Deploy.
4. Dari Add to deployment, pilih Create new deployment, lalu pilih Next.
5. Di halaman Tentukan target, lakukan hal berikut:
   a. Di kotak Nama, masukkan**`Deployment for MyGreengrassCore`**.
   b. Untuk target Deployment, pilih Perangkat inti, dan nama AWS IoT benda untuk perangkat inti Anda. Nilai default dalam tutorial ini adalah *`MyGreengrassCore`*.
   c. Pilih Berikutnya.
6. Pada halaman Pilih komponen, di bawah Komponen saya, verifikasi bahwa com.example.HelloWorldkomponen dipilih, dan pilih Berikutnya.
7. Pada halaman Konfigurasi komponen, pilih com.example.HelloWorld, dan lakukan hal berikut:
   a. Pilih Konfigurasi komponen.
   b. Di bawah Pembaruan konfigurasi, di Konfigurasi untuk menggabungkan, masukkan konfigurasi berikut.

```
{
```

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

Pembaruan konfigurasi ini menetapkan parameter `Message` Hello World ke `universe` untuk perangkat dalam deployment ini.

c. Pilih Konfirmasi.

d. Pilih Berikutnya.

8. Pada halaman Konfigurasikan pengaturan lanjutan, simpan pengaturan konfigurasi default tersebut, dan pilih Selanjutnya.
9. Di halaman Tinjau, pilih Deploy.
10. Verifikasi bahwa deployment berhasil diselesaikan. Deployment ini dapat memakan waktu beberapa menit hingga selesai. Periksa log Hello World untuk memverifikasi perubahan. Jalankan perintah berikut pada perangkat inti Greengrass Anda.

    Linux or Unix

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

    Windows Command Prompt (CMD)

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

    PowerShell

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

    Anda akan melihat pesan yang mirip dengan contoh berikut ini.

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

    > **Note**
    >
    > Jika pesan log tidak berubah, deployment gagal atau tidak mencapai perangkat inti. Hal ini dapat terjadi jika perangkat inti Anda tidak tersambung ke internet atau tidak memiliki izin untuk mengambil artefak dari bucket S3 Anda. Jalankan perintah berikut

pada perangkat inti Anda untuk melihat berkas log perangkat inti AWS IoT Greengrass. File ini mencakup log dari layanan deployment perangkat inti Greengrass.

Linux or Unix

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

Windows Command Prompt (CMD)

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

`type`Perintah menulis konten file ke terminal. Jalankan perintah ini beberapa kali untuk mengamati perubahan dalam file.

PowerShell

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

Untuk informasi selengkapnya, lihat Pemecahan masalah AWS IoT Greengrass V2.

Terapkan komponen Anda () AWS CLI

Untuk men-deploy komponen Hello World Anda

1. Pada komputer pengembangan Anda, buat file bernama `hello-world-deployment.json` dan salin JSON berikut ke dalam file. File ini menentukan komponen dan konfigurasi yang akan di-deploy.

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

File konfigurasi ini menentukan untuk men-deploy versi `1.0.0` komponen Hello World yang telah Anda kembangkan dan deploy dalam prosedur sebelumnya. `configurationUpdate` menentukan untuk menggabungkan konfigurasi komponen dalam string yang dikodekan JSON. Pembaruan konfigurasi ini menetapkan parameter `Message` Hello World ke `universe` untuk perangkat dalam deployment ini.

2. Jalankan perintah berikut untuk men-deploy komponen ke perangkat inti Greengrass Anda. Anda dapat men-deploy ke objek, yang merupakan perangkat individual, atau grup objek, yang merupakan grup perangkat. Ganti *`MyGreengrassCore`*dengan nama AWS IoT benda untuk perangkat inti Anda.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

   Perintah ini menghasilkan respons yang mirip dengan contoh berikut.

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

3. Verifikasi bahwa deployment berhasil diselesaikan. Deployment ini dapat memakan waktu beberapa menit hingga selesai. Periksa log Hello World untuk memverifikasi perubahan. Jalankan perintah berikut pada perangkat inti Greengrass Anda.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

   Anda akan melihat pesan yang mirip dengan contoh berikut ini.

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

   > **Note**
   >
   > Jika pesan log tidak berubah, deployment gagal atau tidak mencapai perangkat inti. Hal ini dapat terjadi jika perangkat inti Anda tidak tersambung ke internet atau tidak memiliki izin untuk mengambil artefak dari bucket S3 Anda. Jalankan perintah berikut pada perangkat inti Anda untuk melihat berkas log perangkat inti AWS IoT Greengrass. File ini mencakup log dari layanan deployment perangkat inti Greengrass.
   >
   > Linux or Unix
   >
   > ```
   > sudo tail -f /greengrass/v2/logs/greengrass.log
   > ```
   >
   > Windows Command Prompt (CMD)
   >
   > ```
   > type C:\greengrass\v2\\logs\\greengrass.log
   > ```

`type`Perintah menulis konten file ke terminal. Jalankan perintah ini beberapa kali untuk mengamati perubahan dalam file.

PowerShell

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

Lihat informasi yang lebih lengkap di Pemecahan masalah AWS IoT Greengrass V2.

Langkah selanjutnya

Anda telah menyelesaikan tutorial ini. Perangkat lunak inti AWS IoT Greengrass dan komponen Hello World Anda berjalan di perangkat Anda. Selain itu, komponen Hello World tersedia di layananAWS IoT Greengrass cloud untuk di-deploy ke perangkat lain. Untuk informasi selengkapnya tentang topik yang dieksplorasi pada tutorial ini, lihat berikut ini:

- Buat AWS IoT Greengrass komponen
- Publikasikan komponen untuk diterapkan ke perangkat inti Anda
- Deploy komponen AWS IoT Greengrass ke perangkat

Menyiapkan perangkat AWS IoT Greengrass inti

Selesaikan tugas di bagian ini untuk menginstal, mengkonfigurasi, dan menjalankan perangkat lunak AWS IoT Greengrass Core.

> **Note**
>
> Bagian ini menjelaskan instalasi lanjutan dan konfigurasi perangkat lunak AWS IoT Greengrass Core. Jika Anda adalah pengguna pertama kali AWS IoT Greengrass V2, kami sarankan Anda terlebih dahulu menyelesaikan tutorial memulai untuk menyiapkan perangkat inti dan menjelajahi fitur-fiturnya. AWS IoT Greengrass

Platform dan persyaratan yang didukung

Sebelum Anda mulai, pastikan Anda memenuhi persyaratan berikut untuk menginstal dan menjalankan perangkat lunak AWS IoT Greengrass Core.

> **Tip**
>
> Anda dapat mencari perangkat yang memenuhi syarat AWS IoT Greengrass V2 di Katalog Perangkat AWS Mitra.

Topik

- Platform yang didukung
- Persyaratan perangkat
- Persyaratan fungsi Lambda

Platform yang didukung

AWS IoT Greengrass secara resmi mendukung perangkat yang menjalankan platform berikut. Perangkat dengan platform yang tidak termasuk dalam daftar ini mungkin berfungsi, tetapi AWS IoT Greengrass pengujian hanya pada platform yang ditentukan ini.

Linux

Arsitektur:

- Armv7l
- Armv8 (AArch64)
- x86_64

Windows

Arsitektur:

- x86_64

Versi:

- Windows 10
- Windows 11
- Windows Server 2019
- Windows Server 2022

> **Note**
>
> Beberapa AWS IoT Greengrass fitur saat ini tidak didukung pada perangkat Windows. Untuk informasi selengkapnya, lihat Kompatibilitas fitur Greengrass oleh sistem operasi dan Pertimbangan fitur untuk perangkat Windows.

Platform Linux juga dapat berjalan AWS IoT Greengrass V2 dalam wadah Docker. Untuk informasi selengkapnya, lihat Jalankan perangkat lunak AWS IoT Greengrass Core dalam wadah Docker.

Untuk membangun sistem operasi berbasis Linux kustom, Anda dapat menggunakan BitBake resep untuk AWS IoT Greengrass V2 dalam proyek. `meta-aws` `meta-aws`Proyek ini menyediakan resep yang dapat Anda gunakan untuk membangun kemampuan perangkat lunak AWS edge dalam sistem Linux tertanam yang dibangun dengan OpenEmbeddeddan kerangka kerja Yocto Project build. Proyek Yocto adalah proyek kolaborasi open source yang membantu Anda membangun sistem berbasis Linux khusus untuk aplikasi tertanam terlepas dari arsitektur perangkat keras. BitBake

Resep untuk AWS IoT Greengrass V2 menginstal, mengonfigurasi, dan secara otomatis menjalankan perangkat lunak AWS IoT Greengrass Core di perangkat Anda.

Persyaratan perangkat

Perangkat harus memenuhi persyaratan berikut untuk menginstal dan menjalankan perangkat lunak AWS IoT Greengrass Core v2.x.

> **Note**
>
> Anda dapat menggunakannya AWS IoT Device Tester AWS IoT Greengrass untuk memverifikasi bahwa perangkat Anda dapat menjalankan perangkat lunak AWS IoT Greengrass Core dan berkomunikasi dengan perangkat lunak AWS Cloud. Untuk informasi selengkapnya, lihat Menggunakan AWS IoT Device Tester untuk AWS IoT Greengrass V2.

Linux

- Penggunaan sebuah Wilayah AWSyang mendukung AWS IoT Greengrass V2. Untuk daftar Wilayah yang didukung, lihat AWS IoT Greengrass V2 titik akhir dan kuota di. Referensi Umum AWS
- Ruang disk minimal 256 MB tersedia untuk perangkat lunak AWS IoT Greengrass Core. Persyaratan ini tidak termasuk komponen yang di-deploy ke perangkat inti.
- Minimum 96 MB RAM dialokasikan untuk perangkat lunak AWS IoT Greengrass Core. Persyaratan ini tidak termasuk komponen yang berjalan pada perangkat inti. Untuk informasi selengkapnya, lihat Kontrol alokasi memori dengan opsi JVM.
- Java Runtime Environment (JRE) versi 8 atau lebih tinggi. Java harus tersedia pada variabel lingkungan PATH pada perangkat. Untuk menggunakan Java untuk mengembangkan komponen kustom, Anda harus menginstal Java Development Kit (JDK). Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan.
- Pustaka GNU C (glibc) versi 2.25 atau lebih besar.
- Anda harus menjalankan perangkat lunak AWS IoT Greengrass Core sebagai pengguna root. Gunakan `sudo`, misalnya.
- Pengguna root yang menjalankan perangkat lunak AWS IoT Greengrass Core, sepertiroot, harus memiliki izin untuk menjalankan `sudo` dengan pengguna dan grup apa pun. File `/etc/`

`sudoers` harus memberikan izin kepada pengguna ini untuk menjalankan `sudo` sebagai grup lain. Izin untuk pengguna di `/etc/sudoers` seharusnya terlihat seperti contoh berikut.

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

- Perangkat inti harus dapat melakukan permintaan keluar ke satu set titik akhir dan port. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.
- Direktori `/tmp` harus dipasang dengan izin `exec`.
- Semua perintah shell berikut:
  - `ps -ax -o pid,ppid`
  - `sudo`
  - `sh`
  - `kill`
  - `cp`
  - `chmod`
  - `rm`
  - `ln`
  - `echo`
  - `exit`
  - `id`
  - `uname`
  - `grep`
- Perangkat Anda mungkin juga memerlukan perintah shell opsional berikut:
  - (Opsional)`systemctl`. Perintah ini digunakan untuk mengatur perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem.
  - (Opsional)`useradd`,`groupadd`, dan`usermod`. Perintah ini digunakan untuk mengatur pengguna `ggc_user` sistem dan grup `ggc_group` sistem.
  - (Opsional)`mkfifo`. Perintah ini digunakan untuk menjalankan fungsi Lambda sebagai komponen.
- Untuk mengonfigurasi batas sumber daya sistem untuk proses komponen, perangkat Anda harus menjalankan kernel Linux versi 2.6.24 atau yang lebih baru.
- Untuk menjalankan fungsi Lambda, perangkat Anda harus memenuhi persyaratan tambahan. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.

Windows

- Penggunaan sebuah Wilayah AWSyang mendukung AWS IoT Greengrass V2. Untuk daftar Wilayah yang didukung, lihat AWS IoT Greengrass V2 titik akhir dan kuota di. Referensi Umum AWS
- Ruang disk minimal 256 MB tersedia untuk perangkat lunak AWS IoT Greengrass Core. Persyaratan ini tidak termasuk komponen yang di-deploy ke perangkat inti.
- Minimum 160 MB RAM dialokasikan untuk perangkat lunak AWS IoT Greengrass Core. Persyaratan ini tidak termasuk komponen yang berjalan pada perangkat inti. Untuk informasi selengkapnya, lihat Kontrol alokasi memori dengan opsi JVM.
- Java Runtime Environment (JRE) versi 8 atau lebih tinggi. Java harus tersedia pada variabel sistem PATH pada perangkat. Untuk menggunakan Java untuk mengembangkan komponen kustom, Anda harus menginstal Java Development Kit (JDK). Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan..

  > **Note**
  >
  > Untuk menggunakan versi 2.5.0 dari inti Greengrass, Anda harus menggunakan Java Runtime Environment (JRE) versi 64-bit. Greengrass nucleus versi 2.5.1 mendukung JRE 32-bit dan 64-bit.

- Pengguna yang menginstal perangkat lunak AWS IoT Greengrass Core harus menjadi administrator.
- Anda harus menginstal perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem. Tentukan `--setup-system-service true` kapan Anda menginstal perangkat lunak.
- Setiap pengguna yang menjalankan proses komponen harus ada di LocalSystem akun, dan nama dan kata sandi pengguna harus ada di instance Credential Manager untuk LocalSystem akun tersebut. Anda dapat mengatur pengguna ini ketika Anda mengikuti petunjuk untuk menginstal perangkat lunak AWS IoT Greengrass Core.
- Perangkat inti harus dapat melakukan permintaan keluar ke satu set titik akhir dan port. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Persyaratan fungsi Lambda

Perangkat Anda harus memenuhi persyaratan berikut untuk menjalankan fungsi Lambda:

- Sebuah sistem operasi berbasis Linux.
- Perangkat Anda harus memiliki perintah shell `mkfifo`.
- Perangkat Anda harus menjalankan pustaka bahasa pemrograman yang dibutuhkan fungsi Lambda. Anda harus menginstal pustaka yang diperlukan pada perangkat dan menambahkannya ke variabel lingkungan `PATH`. Greengrass mendukung semua versi runtime Python, Node.js, dan Java yang didukung Lambda. Greengrass tidak menerapkan batasan tambahan apa pun pada versi runtime Lambda yang tidak digunakan lagi. Untuk informasi selengkapnya tentang AWS IoT Greengrass dukungan untuk runtime Lambda, lihat. Jalankan fungsi AWS Lambda
- Untuk menjalankan fungsi Lambda yang terkontainerisasi, perangkat Anda harus memenuhi persyaratan berikut:
  - Linux kernel versi 4.4 or yag lebih baru.
  - Kernel harus mendukung cgroups v1, dan Anda harus mengaktifkan dan me-mount cgroups berikut:
    - Cgroup memori untuk mengatur batas memori AWS IoT Greengrass untuk fungsi Lambda kontainer.
    - Cgroup perangkat untuk fungsi Lambda dalam kontainer untuk mengakses perangkat atau volume sistem.

    Perangkat lunak AWS IoT Greengrass Core tidak mendukung cgroups v2.

    Untuk memenuhi persyaratan ini, boot perangkat dengan parameter kernel Linux berikut.

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

    > **Tip**
    >
    > Pada Raspberry Pi, edit `/boot/cmdline.txt` file untuk mengatur parameter kernel perangkat.

  - Anda harus mengaktifkan konfigurasi kernel Linux di perangkat:
    - Namespace:
      - `CONFIG_IPC_NS`
      - `CONFIG_UTS_NS`
      - `CONFIG_USER_NS`
      - `CONFIG_PID_NS`

- Cgroup:
  - `CONFIG_CGROUP_DEVICE`
  - `CONFIG_CGROUPS`
  - `CONFIG_MEMCG`
- Lainnya:
  - `CONFIG_POSIX_MQUEUE`
  - `CONFIG_OVERLAY_FS`
  - `CONFIG_HAVE_ARCH_SECCOMP_FILTER`
  - `CONFIG_SECCOMP_FILTER`
  - `CONFIG_KEYS`
  - `CONFIG_SECCOMP`
  - `CONFIG_SHMEM`

> **Tip**
>
> Periksa dokumentasi untuk distribusi Linux Anda untuk mempelajari cara memverifikasi dan mengatur parameter kernel Linux. Anda juga dapat menggunakannya AWS IoT Device Tester AWS IoT Greengrass untuk memverifikasi bahwa perangkat Anda memenuhi persyaratan ini. Untuk informasi selengkapnya, lihat Menggunakan AWS IoT Device Tester untuk AWS IoT Greengrass V2.

Pertimbangan fitur untuk perangkat Windows

Beberapa AWS IoT Greengrass fitur saat ini tidak didukung pada perangkat Windows. Tinjau perbedaan fitur untuk mengonfirmasi apakah perangkat Windows memenuhi kebutuhan Anda. Untuk informasi selengkapnya, lihat Kompatibilitas fitur Greengrass oleh sistem operasi.

Mengatur sebuah Akun AWS

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

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.

2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

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

Untuk membuat pengguna administrator, pilih salah satu opsi berikut.

Pilih salah satu cara untuk mengelola administrator Anda	Untuk	Oleh	Anda juga bisa
Di Pusat Identitas IAM (Direkomendasikan)	Gunakan kredensi jangka pendek untuk mengakses. AWS Ini sejalan dengan praktik terbaik keamanan. Untuk informasi tentang praktik terbaik, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.	Mengikuti petunjuk di Memulai di Panduan AWS IAM Identity Center Pengguna.	Konfigurasikan akses terprogram dengan Mengonfigurasi AWS CLI yang akan digunakan AWS IAM Identity Center dalam AWS Command Line Interface Panduan Pengguna.

Pilih salah satu cara untuk mengelola administrator Anda	Untuk	Oleh	Anda juga bisa
Di IAM (Tidak direkomendasikan)	Gunakan kredensi jangka panjang untuk mengakses. AWS	Mengikuti petunjuk dalam Membuat pengguna admin IAM pertama Anda dan grup pengguna di Panduan Pengguna IAM.	Konfigurasikan akses terprogram dengan Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Instal perangkat lunak inti AWS IoT Greengrass

AWS IoT Greengrass memperluas AWS ke perangkat edge agar dapat bertindak pada data yang dihasilkannya, sementara perangkat tersebut menggunakan AWS Cloud untuk manajemen, analitik, dan penyimpanan yang tahan lama. Instal perangkat lunak inti AWS IoT Greengrass pada perangkat edge untuk terintegrasi dengan AWS IoT Greengrass dan AWS Cloud.

> **Important**
>
> Sebelum Anda mengunduh dan menginstal perangkat lunak AWS IoT Greengrass Core, periksa apakah perangkat inti Anda memenuhi persyaratan untuk menginstal dan menjalankan perangkat lunak AWS IoT Greengrass Core v2.0.

Perangkat lunak inti AWS IoT Greengrass mencakup penginstal yang mengatur perangkat Anda sebagai perangkat inti Greengrass. Saat Anda menjalankan penginstal, Anda dapat mengonfigurasi opsi, seperti folder root dan yang akan Wilayah AWS digunakan. Anda dapat memilih untuk memerintahkan installer tersebut membuat AWS IoT yang diperlukan dan sumber daya IAM untuk Anda. Anda juga dapat memilih untuk men-deploy alat pengembangan lokal untuk mengonfigurasi perangkat yang Anda gunakan untuk pengembangan komponen kustom.

Perangkat lunak inti AWS IoT Greengrass memerlukan AWS IoT berikut dan sumber daya IAM untuk terhubung ke AWS Cloud dan mengoperasikan:

- Objek AWS IoT. Saat Anda mendaftarkan perangkat sebagai objek AWS IoT, perangkat itu dapat menggunakan sertifikat digital untuk diautentikasi dengan AWS. Sertifikat ini memungkinkan perangkat untuk berkomunikasi dengan AWS IoT dan AWS IoT Greengrass. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.
- (Opsional) Grup objek AWS IoT. Anda menggunakan grup objek untuk mengelola armada perangkat inti Greengrass. Saat men-deploy komponen perangkat lunak ke perangkat, Anda dapat memilih untuk men-deploy ke perangkat individual atau ke grup perangkat. Anda dapat menambahkan suatu perangkat ke grup objek untuk men-deploy komponen perangkat lunak grup objek tersebut ke perangkat. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat.
- IAM role. Perangkat inti Greengrass menggunakan penyedia kredensial AWS IoT Core untuk mengotorisasi panggilan ke layanan AWS dengan IAM role. Peran ini memungkinkan perangkat Anda berinteraksiAWS IoT, mengirim log ke Amazon CloudWatch Logs, dan mengunduh artefak komponen khusus dari Amazon Simple Storage Service (Amazon S3). Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.
- Alias peran AWS IoT. Perangkat inti Greengrass menggunakan alias peran untuk mengidentifikasi IAM role yang akan digunakan. Alias peran memungkinkan Anda mengubah IAM role tetapi menjaga konfigurasi perangkat tetap sama. Untuk informasi selengkapnya, lihat Mengotorisasi panggilan langsung ke layanan AWS di Panduan Developer AWS IoT Core.

Pilih salah satu opsi berikut untuk menginstal perangkat lunak inti AWS IoT Greengrass pada perangkat inti Anda.

- Instalasi cepat

  Pilih opsi ini untuk mengatur perangkat inti Greengrass dalam beberapa langkah sesedikit mungkin. Installer tersebut membuat AWS IoT dan sumber daya IAM yang diperlukan untuk Anda. Opsi ini mengharuskan Anda untuk memberikan kredensial AWS ke installer untuk membuat sumber daya di Akun AWS.

  Anda tidak dapat menggunakan opsi ini untuk menginstal di belakang firewall atau proxy jaringan. Jika perangkat Anda berada di belakang firewall atau proxy jaringan, pertimbangkan instalasi manual.

Untuk informasi selengkapnya, lihat Instal perangkat lunak inti AWS IoT Greengrass dengan penyediaan sumber daya otomatis.

- Instalasi manual

  Pilih opsi ini untuk membuat sumber daya AWS yang diperlukan secara manual atau untuk menginstal di belakang firewall atau jaringan proksi. Dengan menggunakan instalasi manual, Anda tidak perlu memberikan izin kepada penginstal untuk membuat sumber daya di Akun AWS, karena Anda membuat AWS IoT dan sumber daya IAM yang diperlukan. Anda juga dapat mengonfigurasi perangkat untuk tersambung ke port 443 atau melalui proksi jaringan. Anda juga dapat mengonfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan kunci pribadi dan sertifikat yang Anda simpan dalam modul keamanan perangkat keras (HSM), Modul Platform Tepercaya (TPM), atau elemen kriptografi lainnya.

  Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya manual.

- Instalasi dengan penyediaan AWS IoT armada

  Pilih opsi ini untuk membuat AWS sumber daya yang diperlukan dari templat penyediaan AWS IoT armada. Anda dapat memilih opsi ini untuk membuat perangkat serupa di armada, atau jika Anda membuat perangkat yang kemudian diaktifkan pelanggan Anda, seperti kendaraan atau perangkat rumah pintar. Perangkat menggunakan sertifikat klaim untuk mengautentikasi dan menyediakan AWS sumber daya, termasuk sertifikat klien X.509 yang digunakan perangkat untuk menyambung ke operasi normal. AWS Cloud Anda dapat menyematkan atau mem-flash sertifikat klaim ke perangkat keras perangkat selama pembuatan, dan Anda dapat menggunakan sertifikat klaim dan kunci yang sama untuk menyediakan beberapa perangkat. Anda juga dapat mengonfigurasi perangkat untuk terhubung pada port 443 atau melalui proxy jaringan.

  Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan AWS IoT armada.

- Instalasi dengan penyediaan khusus

  Pilih opsi ini untuk mengembangkan aplikasi Java kustom yang menyediakan AWS sumber daya yang diperlukan. Anda dapat memilih opsi ini jika Anda membuat sertifikat klien X.509 Anda sendiri atau jika Anda ingin lebih banyak kontrol atas proses penyediaan. AWS IoT Greengrassmenyediakan antarmuka yang dapat Anda terapkan untuk bertukar informasi antara aplikasi penyediaan kustom Anda dan penginstal perangkat lunak AWS IoT Greengrass Core.

Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya khusus.

AWS IoT Greengrass juga menyediakan lingkungan dalam kontainer yang menjalankan perangkat lunak inti AWS IoT Greengrass. Anda dapat menggunakan Dockerfile untuk menjalankan AWS IoT Greengrass dalam kontainer Docker.

Topik

Instal perangkat lunak inti AWS IoT Greengrass dengan penyediaan sumber daya otomatis

Perangkat lunak inti AWS IoT Greengrass mencakup penginstal yang mengatur perangkat Anda sebagai perangkat inti Greengrass. Untuk menyiapkan perangkat dengan cepat, penginstal dapat menyediakan objek AWS IoT, grup objek AWS IoT, IAM role, dan alias peran AWS IoT yang diperlukan perangkat inti untuk beroperasi. Installer juga dapat men-deploy alat pengembangan lokal ke perangkat inti, sehingga Anda dapat menggunakan perangkat tersebut untuk mengembangkan dan menguji komponen perangkat lunak kustom. Installer membutuhkan kredensial AWS untuk menyediakan sumber daya ini dan membuat deployment.

Jika Anda tidak dapat memberikan kredensial AWS untuk perangkat tersebut, Anda dapat menyediakan sumber daya AWS yang dibutuhkan perangkat inti untuk beroperasi. Anda juga dapat men-deploy alat pengembangan ke perangkat inti untuk digunakan sebagai perangkat pengembangan. Hal ini memungkinkan Anda untuk memberikan lebih sedikit izin untuk perangkat ketika Anda menjalankan installer. Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya manual.

> **Important**
>
> Sebelum Anda mengunduh perangkat lunak inti AWS IoT Greengrass, periksa apakah perangkat inti Anda memenuhi persyaratan untuk menginstal dan menjalankan perangkat lunak inti v2.0 AWS IoT Greengrass.

Topik

Mengatur lingkungan perangkat

Ikuti langkah-langkah di bagian ini untuk menyiapkan perangkat Linux atau Windows untuk digunakan sebagai perangkat AWS IoT Greengrass inti Anda.

Siapkan perangkat Linux

Untuk mengatur perangkat Linux untuk AWS IoT Greengrass V2

1. Pasang waktu aktif Java, yang diperlukan oleh perangkat lunak inti AWS IoT Greengrass agar berjalan. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan. Perintah berikut menunjukkan cara menginstal OpenJDK di perangkat Anda.

   - Untuk distribusi berbasis Debian atau berbasis Ubuntu:

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

   - Untuk distribusi berbasis Red Hat:

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

   - Untuk Amazon Linux 2:

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

- Untuk Amazon Linux 2023:

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

Ketika instalasi selesai, jalankan perintah berikut untuk memverifikasi bahwa Java berjalan pada perangkat Linux Anda.

```
java -version
```

Perintah mencetak versi Java yang berjalan pada perangkat. Misalnya, pada distribusi berbasis Debian, output mungkin terlihat mirip dengan sampel berikut.

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

2. (Opsional) Buat pengguna dan grup sistem default yang menjalankan komponen pada perangkat. Anda juga dapat memilih untuk membiarkan penginstal perangkat lunak AWS IoT Greengrass Core membuat pengguna dan grup ini selama instalasi dengan argumen `--component-default-user` installer. Untuk informasi selengkapnya, lihat Argumen penginstal.

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

3. Verifikasi bahwa pengguna yang menjalankan perangkat lunak AWS IoT Greengrass Core (biasanya`root`), memiliki izin untuk menjalankan `sudo` dengan pengguna dan grup apa pun.

   a. Jalankan perintah berikut untuk membuka `/etc/sudoers` file.

   ```
   sudo visudo
   ```

   b. Verifikasi bahwa izin untuk pengguna terlihat seperti contoh berikut.

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

4. (Opsional) Untuk menjalankan fungsi Lambda kontainer, Anda harus mengaktifkan cgroups v1, dan Anda harus mengaktifkan dan memasang memori dan perangkat cgroups. Jika Anda tidak berencana untuk menjalankan fungsi Lambda kontainer, Anda dapat melewati langkah ini.

Untuk mengaktifkan opsi cgroups ini, boot perangkat dengan parameter kernel Linux berikut.

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

Untuk informasi tentang melihat dan menyetel parameter kernel untuk perangkat Anda, lihat dokumentasi untuk sistem operasi dan boot loader. Ikuti instruksi untuk mengatur parameter kernel secara permanen.

5. Instal semua dependensi lain yang diperlukan pada perangkat Anda seperti yang ditunjukkan oleh daftar persyaratan di. Persyaratan perangkat

Siapkan perangkat Windows

> **Note**
>
> Fitur ini tersedia untuk v2.5.0 dan yang lebih baru dari komponen inti Greengrass.

Untuk mengatur perangkat Windows untuk AWS IoT Greengrass V2

1. Pasang waktu aktif Java, yang diperlukan oleh perangkat lunak inti AWS IoT Greengrass agar berjalan. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan.
2. Periksa apakah Java tersedia pada variabel sistem PATH, dan tambahkan jika tidak. LocalSystem Akun menjalankan perangkat lunak AWS IoT Greengrass Core, jadi Anda harus menambahkan Java ke variabel sistem PATH alih-alih variabel pengguna PATH untuk pengguna Anda. Lakukan hal-hal berikut:
   a. Tekan tombol Windows untuk membuka menu mulai.
   b. Ketik **`environment variables`** untuk mencari opsi sistem dari menu mulai.
   c. Di hasil pencarian menu mulai, pilih Edit variabel lingkungan sistem untuk membuka jendela Properti sistem.
   d. Pilih variabel Lingkungan... untuk membuka jendela Variabel Lingkungan.
   e. Di bawah Variabel sistem, pilih Path, lalu pilih Edit. Di jendela variabel Edit lingkungan, Anda dapat melihat setiap jalur pada baris terpisah.

f. Periksa apakah jalur ke `bin` folder instalasi Java ada. Jalannya mungkin terlihat mirip dengan contoh berikut.

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

g. Jika `bin` folder instalasi Java hilang dari Path, pilih Baru untuk menambahkannya, lalu pilih OK.

3. Buka Windows Command Prompt (`cmd.exe`) sebagai administrator.
4. Buat pengguna default di LocalSystem akun di perangkat Windows. Ganti *`kata sandi`* dengan kata sandi yang aman.

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

> **Tip**
>
> Bergantung pada konfigurasi Windows Anda, kata sandi pengguna mungkin diatur untuk kedaluwarsa pada tanggal di masa mendatang. Untuk memastikan aplikasi Greengrass Anda terus beroperasi, lacak kapan kata sandi kedaluwarsa, dan perbarui sebelum kedaluwarsa. Anda juga dapat mengatur kata sandi pengguna agar tidak pernah kedaluwarsa.
>
> - Untuk memeriksa kapan pengguna dan kata sandinya kedaluwarsa, jalankan perintah berikut.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Untuk mengatur kata sandi pengguna agar tidak pernah kedaluwarsa, jalankan perintah berikut.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Jika Anda menggunakan Windows 10 atau yang lebih baru di mana wmicperintah tidak digunakan lagi, jalankan perintah berikut. PowerShell
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

5. Unduh dan instal PsExecutilitas dari Microsoft pada perangkat.

6. Gunakan PsExec utilitas untuk menyimpan nama pengguna dan kata sandi untuk pengguna default dalam contoh Credential Manager untuk LocalSystem akun tersebut. Ganti *kata sandi* dengan kata sandi pengguna yang Anda tetapkan sebelumnya.

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

Jika PsExec License Agreementterbuka, pilih Acceptuntuk menyetujui lisensi dan jalankan perintah.

> **Note**
>
> Pada perangkat Windows, LocalSystem akun menjalankan inti Greengrass, dan Anda harus menggunakan utilitas untuk menyimpan PsExec informasi pengguna default di akun. LocalSystem Menggunakan aplikasi Credential Manager menyimpan informasi ini di akun Windows dari pengguna yang saat ini masuk, bukan LocalSystem akun.

Berikan kredensial AWS ke perangkat

Berikan kredensial AWS ke perangkat Anda agar installer dapat menyediakan sumber daya AWS yang diperlukan. Untuk informasi lebih lanjut tentang izin yang diperlukan, lihat Kebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya.

Untuk memberikan kredensial AWS ke perangkat

- Berikan AWS kredensi Anda ke perangkat sehingga penginstal dapat menyediakan sumber daya AWS IoT dan IAM untuk perangkat inti Anda. Untuk meningkatkan keamanan, sebaiknya Anda mendapatkan kredensi sementara untuk peran IAM yang hanya mengizinkan izin minimum yang diperlukan untuk penyediaan. Untuk informasi selengkapnya, lihat Kebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya.

  > **Note**
  >
  > Penginstal tidak menyimpan atau menyimpan kredensial Anda.

  Di perangkat Anda, lakukan salah satu hal berikut untuk mengambil kredensi dan membuatnya tersedia untuk penginstal perangkat lunak AWS IoT Greengrass Core:

- (Disarankan) Gunakan kredenal temporay dari AWS IAM Identity Center

  a. Berikan ID kunci akses, kunci akses rahasia, dan token sesi dari Pusat Identitas IAM. Untuk informasi selengkapnya, lihat Penyegaran kredenal manual di Mendapatkan dan menyegarkan kredenal sementara di panduan pengguna Pusat Identitas IAM.

  b. Jalankan perintah berikut untuk memberikan kredensial ke perangkat lunak inti AWS IoT Greengrass.

  Linux or Unix

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

  Windows Command Prompt (CMD)

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

  PowerShell

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

- Gunakan kredenal keamanan sementara dari peran IAM:

  a. Berikan access key ID, secret access key, dan token sesi dari IAM role yang Anda teruskan. Untuk informasi selengkapnya tentang cara mengambil kredenal ini, lihat Meminta kredenal keamanan sementara di Panduan Pengguna IAM.

  b. Jalankan perintah berikut untuk memberikan kredensial ke perangkat lunak inti AWS IoT Greengrass.

  Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

- Gunakan kredensial jangka panjang dari pengguna IAM:

  a. Berikan access key ID dan secret access key untuk pengguna IAM Anda. Anda dapat membuat pengguna IAM untuk penyediaan yang kemudian Anda hapus. Untuk kebijakan IAM untuk memberikan pengguna, lihatKebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya. Untuk informasi selengkapnya tentang cara mengambil kredensial jangka panjang, lihat Mengelola access key untuk pengguna IAM di Panduan Pengguna IAM.

  b. Jalankan perintah berikut untuk memberikan kredensial ke perangkat lunak inti AWS IoT Greengrass.

  Linux or Unix

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

  Windows Command Prompt (CMD)

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

  PowerShell

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

c. (Opsional) Jika Anda membuat pengguna IAM untuk menyediakan perangkat Greengrass Anda, hapus pengguna tersebut.

d. (Opsional) Jika Anda menggunakan ID kunci akses dan kunci akses rahasia dari pengguna IAM yang ada, perbarui kunci untuk pengguna sehingga tidak lagi valid. Untuk informasi selengkapnya, lihat Memperbarui kunci akses di panduan AWS Identity and Access Management pengguna.

Unduh perangkat lunak inti AWS IoT Greengrass

Anda dapat mengunduh versi terbaru perangkat lunak inti AWS IoT Greengrass dari lokasi berikut:

- https://d2s8p88vqu9w66.cloudfront.net/releases/ greengrass-nucleus-latest .zip

> **Note**
>
> Anda dapat mengunduh versi tertentu perangkat lunak inti AWS IoT Greengrass dari lokasi berikut. Ganti *versi* dengan versi yang akan diunduh.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Untuk mengunduh perangkat lunak AWS IoT Greengrass Core

1. Di perangkat inti Anda, unduh perangkat lunak AWS IoT Greengrass Core ke file bernama`greengrass-nucleus-latest.zip`.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

PowerShell

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

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

2. (Opsional) Untuk memverifikasi tanda tangan perangkat lunak inti Greengrass

> **Note**
>
> Fitur ini tersedia dengan Greengrass nucleus versi 2.9.5 dan yang lebih baru.

a. Gunakan perintah berikut untuk memverifikasi tanda tangan artefak inti Greengrass Anda:

Linux or Unix

```
jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
```

Windows Command Prompt (CMD)

Nama file mungkin terlihat berbeda tergantung pada versi JDK yang Anda instal. Ganti *jdk17.0.6_10* dengan versi JDK yang Anda instal.

```
"C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
verify -certs -verbose greengrass-nucleus-latest.zip
```

PowerShell

Nama file mungkin terlihat berbeda tergantung pada versi JDK yang Anda instal. Ganti *jdk17.0.6_10* dengan versi JDK yang Anda instal.

```
'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
verify -certs -verbose greengrass-nucleus-latest.zip
```

b. `jarsigner` Pemanggilan menghasilkan output yang menunjukkan hasil verifikasi.

i. Jika file zip inti Greengrass ditandatangani, output berisi pernyataan berikut:

```
jar verified.
```

ii. Jika file zip inti Greengrass tidak ditandatangani, output berisi pernyataan berikut:

```
jar is unsigned.
```

c. Jika Anda memberikan `-certs` opsi Jarsigner bersama dengan `-verify` dan `-verbose` opsi, output juga menyertakan informasi sertifikat penandatangan terperinci.

3. Unzip perangkat lunak inti AWS IoT Greengrass ke folder di perangkat Anda. Ganti *`GreengrassInstaller`*dengan folder yang ingin Anda gunakan.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

4. (Opsional) Jalankan perintah berikut untuk melihat versi perangkat lunak inti AWS IoT Greengrass.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

> **Important**
>
> Jika Anda menginstal versi inti Greengrass lebih awal dari v2.4.0, jangan hapus folder ini setelah Anda menginstal perangkat lunak Core. AWS IoT Greengrass Perangkat lunak inti AWS IoT Greengrass menggunakan file di folder ini yang akan dijalankan.
> Jika Anda mengunduh versi terbaru perangkat lunak, Anda menginstal v2.4.0 atau yang lebih baru, dan Anda dapat menghapus folder ini setelah Anda menginstal perangkat lunak AWS IoT Greengrass Core.

Instal perangkat lunak inti AWS IoT Greengrass

Jalankan installer dengan argumen yang menentukan untuk melakukan hal berikut:

- Buat sumber daya AWS yang dibutuhkan perangkat inti untuk beroperasi.
- Tentukan untuk menggunakan pengguna `ggc_user` sistem untuk menjalankan komponen perangkat lunak pada perangkat inti. Pada perangkat Linux, perintah ini juga menentukan untuk menggunakan grup `ggc_group` sistem, dan penginstal membuat pengguna dan grup sistem untuk Anda.
- Siapkan perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem yang berjalan saat boot. Pada perangkat Linux, ini membutuhkan sistem init Systemd.

> **Important**
>
> Pada perangkat inti Windows, Anda harus mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Untuk menyiapkan perangkat pengembangan dengan alat pengembangan lokal, tentukan argumen `--deploy-dev-tools true`. Alat pengembangan lokal dapat memakan waktu hingga satu menit untuk ter-deploy setelah instalasi tersebut selesai.

Untuk informasi lebih lanjut tentang argumen yang dapat Anda tentukan, lihat Argumen penginstal.

> **Note**
>
> Jika Anda menjalankan AWS IoT Greengrass pada perangkat dengan memori terbatas, Anda dapat mengontrol jumlah memori yang digunakan oleh perangkat lunak inti AWS

IoT Greengrass. Untuk mengontrol alokasi memori, Anda dapat mengatur pilihan ukuran tumpukan JVM di konfigurasi parameter `jvmOptions` dalam komponen nukleus anda. Untuk informasi selengkapnya, lihat Kontrol alokasi memori dengan opsi JVM.

Untuk menginstal AWS IoT Greengrass perangkat lunak Core

1. Jalankan penginstal inti AWS IoT Greengrass. Ganti nilai-nilai argumen dalam perintah Anda sebagai berikut:

   a. */greengrass/v2*atau *C:\greengrass\v2*: Jalur ke folder root yang akan digunakan untuk menginstal perangkat lunak AWS IoT Greengrass Core.

   b. *GreengrassInstaller*. Path ke folder tempat Anda membongkar penginstal perangkat lunak inti AWS IoT Greengrass.

   c. *region*. Wilayah AWS tempat untuk menemukan atau membuat sumber daya.

   d. *MyGreengrassCore*. Nama objek AWS IoT untuk perangkat inti Greengrass Anda. Jika objek tidak ada, installer akan membuatnya. Penginstal mengunduh sertifikat tersebut untuk diautentikasi sebagai objek AWS IoT. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.

   > **Note**
   >
   > Nama objek tidak dapat berisi karakter titik dua (`:`).

   e. *MyGreengrassCoreGroup*. Nama grup objek AWS IoT untuk perangkat inti Greengrass Anda. Jika grup objek tidak ada, installer akan membuatnya dan menambahkan objek padanya. Jika grup objek ada dan memiliki deployment yang aktif, perangkat inti akan men-download dan menjalankan perangkat lunak yang ditetapkan oleh deployment.

   > **Note**
   >
   > Nama grup objek tidak dapat berisi karakter titik dua (`:`).

   f. *ThingPolicyGreenGrassV2IoT*. Nama AWS IoT kebijakan yang memungkinkan perangkat inti Greengrass untuk berkomunikasi dengan dan. AWS IoT AWS IoT Greengrass Jika AWS IoT kebijakan tidak ada, penginstal akan membuat AWS IoT kebijakan permisif dengan nama ini. Anda dapat membatasi izin kebijakan ini untuk kasus penggunaan Anda.

Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2.

g. *GreenGrassV2 TokenExchangeRole*. Nama IAM role yang memungkinkan perangkat inti Greengrass untuk mendapatkan kredensial AWS sementara. Jika peran itu tidak ada, penginstal akan membuatnya dan membuat serta melampirkan kebijakan bernama *GreengrassV2TokenExchangeRole*Access. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

h. *GreengrassCoreTokenExchangeRoleAlias*. Nama IAM role yang memungkinkan perangkat inti Greengrass untuk mendapatkan kredensial sementara nanti. Jika alias peran tidak ada, penginstal akan membuatnya dan mengarahkannya ke IAM role yang Anda tentukan. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Linux or Unix

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

Windows Command Prompt (CMD)

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

```
  --setup-system-service true
```

PowerShell

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

> **Important**
>
> Pada perangkat inti Windows, Anda harus menentukan `--setup-system-service true` untuk mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Installer mencetak pesan berikut jika berhasil:

- Jika Anda menentukan `--provision`, penginstal akan mencetak `Successfully configured Nucleus with provisioned resource details` jika berhasil mengonfigurasi sumber daya.
- Jika Anda menentukan `--deploy-dev-tools`, penginstal akan mencetak `Configured Nucleus to deploy aws.greengrass.Cli component` jika berhasil membuat deployment.
- Jika Anda menentukan `--setup-system-service true`, penginstal akan mencetak `Successfully set up Nucleus as a system service` jika ia mengatur dan menjalankan perangkat lunak sebagai layanan.
- Jika Anda tidak menentukan `--setup-system-service true`, penginstal akan mencetak `Launched Nucleus successfully` jika berhasil dan menjalankan perangkat lunak.

2. Lewati langkah ini jika Anda menginstal v2.0.4 Inti Greengrass atau yang lebih baru. Jika Anda mengunduh versi terbaru dari perangkat lunak, Anda menginstal v2.0.4 atau yang lebih baru.

Jalankan perintah berikut untuk mengatur izin file yang diperlukan untuk folder root perangkat lunak inti AWS IoT Greengrass Anda. Ganti */greengrass/v2* dengan folder root yang Anda tentukan dalam perintah instalasi Anda, dan ganti */greengrass* dengan folder induk untuk folder root Anda.

```
sudo chmod 755 /greengrass/v2 && sudo chmod 755 /greengrass
```

Jika Anda menginstal perangkat lunak inti AWS IoT Greengrass sebagai layanan sistem, installer akan menjalankan perangkat lunak untuk Anda. Jika tidak, Anda harus menjalankan perangkat lunak itu secara manual. Untuk informasi selengkapnya, lihat Jalankan perangkat lunak inti AWS IoT Greengrass.

> **Note**
>
> Secara default, IAM role yang dibuat oleh installer tidak mengizinkan akses ke artefak komponen dalam Bucket S3. Untuk men-deploy komponen kustom yang menentukan artefak di Amazon S3, Anda harus menambahkan izin untuk peran tersebut untuk memungkinkan perangkat inti Anda untuk mengambil artefak komponen. Untuk informasi selengkapnya, lihat Izinkan akses ke bucket S3 untuk artefak komponen.
> Jika Anda belum memiliki bucket S3 untuk artefak komponen, Anda dapat menambahkan izin ini nanti setelah membuat bucket.

Untuk informasi lebih lanjut tentang cara mengonfigurasi dan menggunakan perangkat lunak dan AWS IoT Greengrass, lihat hal berikut:

- Konfigurasikan perangkat lunak AWS IoT Greengrass Inti
- Kembangkan AWS IoT Greengrass komponen
- Deploy komponen AWS IoT Greengrass ke perangkat
- Antarmuka Baris Perintah Greengrass

Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya manual

Perangkat lunak AWS IoT Greengrass Core mencakup penginstal yang mengatur perangkat Anda sebagai perangkat inti Greengrass. Untuk mengatur perangkat secara manual, Anda dapat membuat sumber daya yang diperlukan AWS IoT dan IAM untuk digunakan perangkat. Jika Anda membuat sumber daya ini secara manual, Anda tidak perlu memberikan AWS kredensi kepada penginstal.

Saat Anda menginstal perangkat lunak AWS IoT Greengrass Core secara manual, Anda juga dapat mengonfigurasi perangkat untuk menggunakan proxy jaringan atau terhubung ke AWS port 443. Anda mungkin perlu menentukan opsi konfigurasi ini jika perangkat berjalan di belakang firewall atau proksi jaringan, misalnya. Untuk informasi selengkapnya, lihat Hubungkan pada port 443 atau melalui proksi jaringan.

Anda juga dapat mengonfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan modul keamanan perangkat keras (HSM) melalui antarmuka PKCS #11. Fitur ini memungkinkan Anda menyimpan file kunci pribadi dan sertifikat dengan aman sehingga tidak terekspos atau digandakan dalam perangkat lunak. Anda dapat menyimpan kunci pribadi dan sertifikat pada modul perangkat keras seperti HSM, Trusted Platform Module (TPM), atau elemen kriptografi lainnya. Fitur ini hanya tersedia di perangkat Linux. Untuk informasi selengkapnya tentang keamanan perangkat keras dan persyaratan untuk menggunakannya, lihatIntegrasi keamanan perangkat keras.

> **Important**
>
> Sebelum Anda mengunduh perangkat lunak AWS IoT Greengrass Core, periksa apakah perangkat inti Anda memenuhi persyaratan untuk menginstal dan menjalankan perangkat lunak AWS IoT Greengrass Core v2.0.

Topik

- Mengatur lingkungan perangkat
- Unduh perangkat lunak AWS IoT Greengrass Inti
- Instal perangkat lunak AWS IoT Greengrass Inti

Ambil titik akhir AWS IoT

Dapatkan AWS IoT titik akhir untuk Anda Akun AWS, dan simpan untuk digunakan nanti. Perangkat Anda menggunakan titik akhir ini untuk tersambung ke AWS IoT. Lakukan hal-hal berikut:

1. Dapatkan titik akhir AWS IoT data untuk Anda Akun AWS.

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

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Dapatkan titik akhir AWS IoT kredensial untuk Anda. Akun AWS

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

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "endpointAddress": "device-credentials-prefix.credentials.iot.us-
   west-2.amazonaws.com"
   }
   ```

Buat AWS IoT sesuatu

AWS IoT hal-hal mewakili perangkat dan entitas logis yang terhubung ke AWS IoT. Perangkat inti Greengrass adalah benda. AWS IoT Saat Anda mendaftarkan perangkat sebagai AWS IoT sesuatu, perangkat tersebut dapat menggunakan sertifikat digital untuk mengautentikasi. AWS

Di bagian ini, Anda membuat AWS IoT sesuatu yang mewakili perangkat Anda.

Untuk menciptakan AWS IoT sesuatu

1. Buat AWS IoT sesuatu untuk perangkat Anda. Pada komputer pengembangan Anda, jalankan perintah berikut.

   - Ganti *MyGreengrassCore* dengan nama benda yang akan digunakan. Nama ini juga merupakan nama perangkat inti Greengrass Anda.

     > **Note**
     >
     > Nama objek tidak dapat berisi karakter titik dua (`:`).

   ```
   aws iot create-thing --thing-name MyGreengrassCore
   ```

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "thingName": "MyGreengrassCore",
     "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
     "thingId": "8cb4b6cd-268e-495d-b5b9-1713d71dbf42"
   }
   ```

2. (Opsional) Tambahkan AWS IoT benda ke grup hal baru atau yang sudah ada. Anda menggunakan grup objek untuk mengelola armada perangkat inti Greengrass. Saat menerapkan komponen perangkat lunak ke perangkat, Anda dapat menargetkan perangkat individual atau grup perangkat. Anda dapat menambahkan suatu perangkat ke grup objek dengan deployment Greengrass aktif untuk men-deploy komponen perangkat lunak grup objek tersebut ke perangkat. Lakukan hal-hal berikut:

   a. (Opsional) Buat grup AWS IoT benda.

      - Ganti *MyGreengrassCoreGroup*dengan nama grup benda yang akan dibuat.

        > **Note**
        >
        > Nama grup objek tidak dapat berisi karakter titik dua (`:`).

```
aws iot create-thing-group --thing-group-name MyGreengrassCoreGroup
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "thingGroupName": "MyGreengrassCoreGroup",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
MyGreengrassCoreGroup",
  "thingGroupId": "4df721e1-ff9f-4f97-92dd-02db4e3f03aa"
}
```

b. Tambahkan AWS IoT benda itu ke grup benda.

- Ganti *MyGreengrassCore* dengan nama AWS IoT barang Anda.
- Ganti *MyGreengrassCoreGroup*dengan nama grup benda.

```
aws iot add-thing-to-thing-group --thing-name MyGreengrassCore --thing-group-
name MyGreengrassCoreGroup
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

Buat sertifikat benda

Saat Anda mendaftarkan perangkat sebagai AWS IoT sesuatu, perangkat tersebut dapat menggunakan sertifikat digital untuk mengautentikasi. AWS Sertifikat ini memungkinkan perangkat untuk berkomunikasi dengan AWS IoT dan AWS IoT Greengrass.

Di bagian ini, Anda membuat dan mengunduh sertifikat yang dapat digunakan perangkat Anda untuk terhubung AWS.

Jika Anda ingin mengonfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan modul keamanan perangkat keras (HSM) untuk menyimpan kunci pribadi dan sertifikat dengan aman, ikuti langkah-langkah untuk membuat sertifikat dari kunci pribadi di HSM. Jika tidak, ikuti langkah-langkah untuk membuat sertifikat dan kunci pribadi dalam AWS IoT layanan. Fitur keamanan perangkat keras hanya tersedia di perangkat Linux. Untuk informasi selengkapnya tentang keamanan perangkat keras dan persyaratan untuk menggunakannya, lihatIntegrasi keamanan perangkat keras.

Buat sertifikat dan kunci pribadi dalam AWS IoT layanan

Untuk membuat sertifikat benda

1. Buat folder tempat Anda mengunduh sertifikat untuk AWS IoT benda itu.

```
mkdir greengrass-v2-certs
```

2. Buat dan unduh sertifikat untuk AWS IoT benda itu.

```
aws iot create-keys-and-certificate --set-as-active --certificate-pem-outfile
 greengrass-v2-certs/device.pem.crt --public-key-outfile greengrass-v2-certs/
public.pem.key --private-key-outfile greengrass-v2-certs/private.pem.key
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
  "certificateId":
 "aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
  "certificatePem": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
  "keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----\
MIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\
MMEXAMPLEuuN/dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/gHr99VEEXAMPLE5VF13\
```

```
59VK7cEXAMPLE67GK+y+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQWE
\\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa\
hJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\
FQIDAQAB\
-----END PUBLIC KEY-----\
",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\
key omitted for security reasons\
-----END RSA PRIVATE KEY-----\
"
  }
}
```

Simpan Nama Sumber Daya Amazon (ARN) sertifikat yang akan digunakan untuk mengonfigurasi sertifikat nanti.

Buat sertifikat dari kunci pribadi di HSM

> **Note**
>
> Fitur ini tersedia untuk v2.5.3 dan yang lebih baru dari komponen inti Greengrass. AWS IoT Greengrass saat ini tidak mendukung fitur ini di perangkat inti Windows.

Untuk membuat sertifikat benda

1. Pada perangkat inti, inisialisasi token PKCS #11 di HSM, dan buat kunci pribadi. Kunci pribadi harus berupa kunci RSA dengan ukuran kunci RSA-2048 (atau lebih besar) atau kunci ECC.

   > **Note**
   >
   > Untuk menggunakan modul keamanan perangkat keras dengan kunci ECC, Anda harus menggunakan Greengrass nucleus v2.5.6 atau yang lebih baru.
   > Untuk menggunakan modul keamanan perangkat keras dan manajer rahasia, Anda harus menggunakan modul keamanan perangkat keras dengan kunci RSA.

Periksa dokumentasi untuk HSM Anda untuk mempelajari cara menginisialisasi token dan menghasilkan kunci pribadi. Jika HSM Anda mendukung ID objek, tentukan ID objek saat Anda membuat kunci pribadi. Simpan ID slot, PIN pengguna, label objek, ID objek (jika HSM Anda menggunakan satu) yang Anda tentukan saat Anda menginisialisasi token dan menghasilkan kunci pribadi. Anda menggunakan nilai-nilai ini nanti ketika Anda mengimpor sertifikat hal ke HSM dan mengkonfigurasi perangkat lunak AWS IoT Greengrass Core.

2. Buat permintaan penandatanganan sertifikat (CSR) dari kunci pribadi. AWS IoT menggunakan CSR ini untuk membuat sertifikat sesuatu untuk kunci pribadi yang Anda buat di HSM. Untuk informasi tentang cara membuat CSR dari kunci pribadi, lihat dokumentasi untuk HSM Anda. CSR adalah file, sepertiiotdevicekey.csr.

3. Salin CSR dari perangkat ke komputer pengembangan Anda. Jika SSH dan SCP diaktifkan pada komputer pengembangan dan perangkat, Anda dapat menggunakan `scp` perintah di komputer pengembangan Anda untuk mentransfer CSR. Ganti *device-ip-address dengan alamat* IP perangkat Anda, dan ganti *~/iotdevicekey.csr* dengan path ke file CSR di perangkat.

```
scp device-ip-address:~/iotdevicekey.csr iotdevicekey.csr
```

4. Di komputer pengembangan Anda, buat folder tempat Anda mengunduh sertifikat untuk AWS IoT benda itu.

```
mkdir greengrass-v2-certs
```

5. Gunakan file CSR untuk membuat dan mengunduh sertifikat untuk AWS IoT hal itu ke komputer pengembangan Anda.

```
aws iot create-certificate-from-csr --set-as-active --certificate-signing-
request=file://iotdevicekey.csr --certificate-pem-outfile greengrass-v2-certs/
device.pem.crt
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
  "certificateId":
 "aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
  "certificatePem": "-----BEGIN CERTIFICATE-----
```

```
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
-----END CERTIFICATE-----"
}
```

Simpan ARN sertifikat untuk digunakan untuk mengonfigurasi sertifikat nanti.

Konfigurasikan sertifikat benda

Lampirkan sertifikat benda ke AWS IoT benda yang Anda buat sebelumnya, dan tambahkan AWS IoT kebijakan ke sertifikat untuk menentukan AWS IoT izin untuk perangkat inti.

Untuk mengkonfigurasi sertifikat benda

1. Lampirkan sertifikat ke AWS IoT benda itu.

   - Ganti *MyGreengrassCore* dengan nama AWS IoT barang Anda.
   - Ganti Amazon Resource Name (ARN) sertifikat dengan ARN sertifikat yang Anda buat pada langkah sebelumnya.

   ```
   aws iot attach-thing-principal --thing-name MyGreengrassCore
    --principal arn:aws:iot:us-west-2:123456789012:cert/
   aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
   ```

   Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

2. Buat dan lampirkan AWS IoT kebijakan yang menentukan AWS IoT izin untuk perangkat inti Greengrass Anda. Kebijakan berikut memungkinkan akses ke semua topik MQTT dan operasi Greengrass, sehingga perangkat Anda bekerja dengan aplikasi kustom dan perubahan di masa mendatang yang memerlukan operasi Greengrass baru. Anda dapat membatasi kebijakan ini berdasarkan kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2.

   Jika Anda telah menyiapkan perangkat inti Greengrass sebelumnya, Anda dapat melampirkan AWS IoT kebijakannya alih-alih membuat yang baru.

   Lakukan hal-hal berikut:

   a. Buat file yang berisi dokumen AWS IoT kebijakan yang dibutuhkan perangkat inti Greengrass.

      Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

      ```
      nano greengrass-v2-iot-policy.json
      ```

      Salin JSON berikut ke dalam file.

      ```
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "iot:Publish",
              "iot:Subscribe",
              "iot:Receive",
              "iot:Connect",
              "greengrass:*"
            ],
            "Resource": [
              "*"
            ]
          }
        ]
      }
      ```

b. Buat AWS IoT kebijakan dari dokumen kebijakan.

- Ganti *GreenGrassV2IoT ThingPolicy* dengan nama kebijakan yang akan dibuat.

```
aws iot create-policy --policy-name GreengrassV2IoTThingPolicy --policy-
document file://greengrass-v2-iot-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "policyName": "GreengrassV2IoTThingPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
  "policyDocument": "{
    \\"Version\\": \\"2012-10-17\\",
    \\"Statement\\": [
      {
        \\"Effect\\": \\"Allow\\",
        \\"Action\\": [
          \\"iot:Publish\\",
          \\"iot:Subscribe\\",
          \\"iot:Receive\\",
          \\"iot:Connect\\",
          \\"greengrass:*\\"
        ],
        \\"Resource\\": [
          \\"*\\"
        ]
      }
    ]
  }",
  "policyVersionId": "1"
}
```

c. Lampirkan AWS IoT kebijakan ke sertifikat AWS IoT benda itu.

- Ganti *GreenGrassV2IoT ThingPolicy* dengan nama kebijakan yang akan dilampirkan.
- Ganti ARN target dengan ARN sertifikat untuk objek AWS IoT Anda.

```
aws iot attach-policy --policy-name GreengrassV2IoTThingPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

Buat peran pertukaran token

Perangkat inti Greengrass menggunakan peran layanan IAM, yang disebut peran pertukaran token, untuk mengotorisasi panggilan ke layanan. AWS Perangkat menggunakan penyedia AWS IoT kredensional untuk mendapatkan AWS kredensi sementara untuk peran ini, yang memungkinkan perangkat berinteraksi, mengirim log ke Amazon Log AWS IoT, dan mengunduh CloudWatch artefak komponen khusus dari Amazon S3. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Anda menggunakan alias AWS IoT peran untuk mengonfigurasi peran pertukaran token untuk perangkat inti Greengrass. Alias peran memungkinkan Anda mengubah peran pertukaran token untuk suatu perangkat tetapi menjaga konfigurasi perangkat tetap sama. Untuk informasi selengkapnya, lihat Mengotorisasi panggilan langsung ke layanan AWS di Panduan Developer AWS IoT Core .

Di bagian ini, Anda membuat peran IAM pertukaran token dan alias AWS IoT peran yang menunjuk ke peran tersebut. Jika Anda telah menyiapkan perangkat inti Greengrass, Anda dapat menggunakan peran pertukaran token dan alias peran alih-alih membuat yang baru. Kemudian, Anda mengonfigurasi objek AWS IoT untuk menggunakan peran dan alias itu.

Buat peran pertukaran token IAM role

1. Buat peran IAM yang dapat digunakan perangkat Anda sebagai peran pertukaran token. Lakukan hal-hal berikut:

   a. Buat file yang berisi dokumen kebijakan kepercayaan yang memerlukan peran pertukaran token.

      Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano device-role-trust-policy.json
```

Salin JSON berikut ke dalam file.

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

b. Buat peran pertukaran token dengan dokumen kebijakan kepercayaan.

- Ganti *GreenGrassV2 TokenExchange Role dengan nama peran* IAM yang akan dibuat.

```
aws iam create-role --role-name GreengrassV2TokenExchangeRole --assume-role-
policy-document file://device-role-trust-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "Role": {
    "Path": "/",
    "RoleName": "GreengrassV2TokenExchangeRole",
    "RoleId": "AROAZ2YMUHYHK5OKM77FB",
    "Arn": "arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole",
    "CreateDate": "2021-02-06T00:13:29+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
```

```
            "Service": "credentials.iot.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
```

c. Buat file yang berisi dokumen kebijakan akses yang diperlukan oleh peran pertukaran token.

Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano device-role-access-policy.json
```

Salin JSON berikut ke dalam file.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Kebijakan akses ini tidak mengizinkan akses ke artefak komponen dalam bucket S3. Untuk men-deploy komponen kustom yang menentukan artefak di Amazon S3, Anda harus menambahkan izin untuk peran tersebut untuk memungkinkan perangkat inti Anda untuk mengambil artefak komponen. Untuk informasi selengkapnya, lihat Izinkan akses ke bucket S3 untuk artefak komponen.

Jika Anda belum memiliki bucket S3 untuk artefak komponen, Anda dapat menambahkan izin ini nanti setelah membuat bucket.

d. Buat kebijakan IAM dari dokumen kebijakan.

- Ganti *GreenGrassV2 TokenExchange RoleAccess* dengan nama kebijakan IAM yang akan dibuat.

```
aws iam create-policy --policy-name GreengrassV2TokenExchangeRoleAccess --
policy-document file://device-role-access-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "Policy": {
    "PolicyName": "GreengrassV2TokenExchangeRoleAccess",
    "PolicyId": "ANPAZ2YMUHYHACI7C5Z66",
    "Arn": "arn:aws:iam::123456789012:policy/
GreengrassV2TokenExchangeRoleAccess",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2021-02-06T00:37:17+00:00",
    "UpdateDate": "2021-02-06T00:37:17+00:00"
  }
}
```

e. Lampirkan kebijakan IAM untuk peran pertukaran token.

- Ganti *GreenGrassV2 TokenExchange Role dengan nama peran* IAM.
- Ganti ARN peran dengan ARN dari kebijakan IAM yang Anda buat di langkah sebelumnya.

```
aws iam attach-role-policy --role-name GreengrassV2TokenExchangeRole --policy-
arn arn:aws:iam::123456789012:policy/GreengrassV2TokenExchangeRoleAccess
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

2. Buat alias AWS IoT peran yang menunjuk ke peran pertukaran token.

   - Ganti *GreengrassCoreTokenExchangeRoleAlias*dengan nama alias peran yang akan dibuat.
   - Ganti ARN peran dengan ARN dari IAM role yang Anda buat di langkah sebelumnya.

   ```
   aws iot create-role-alias --role-alias GreengrassCoreTokenExchangeRoleAlias --role-
   arn arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole
   ```

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "roleAlias": "GreengrassCoreTokenExchangeRoleAlias",
     "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
   GreengrassCoreTokenExchangeRoleAlias"
   }
   ```

   > **Note**
   >
   > Untuk membuat alias peran, Anda harus memiliki izin untuk melewati IAM role pertukaran token ke AWS IoT. Jika Anda menerima pesan galat saat mencoba membuat alias peran, periksa apakah AWS pengguna Anda memiliki izin ini. Untuk informasi selengkapnya, lihat Memberikan izin pengguna untuk meneruskan peran ke AWS layanan di AWS Identity and Access Management Panduan Pengguna.

3. Buat dan lampirkan AWS IoT kebijakan yang memungkinkan perangkat inti Greengrass Anda menggunakan alias peran untuk mengambil peran pertukaran token. Jika Anda telah menyiapkan perangkat inti Greengrass sebelumnya, Anda dapat melampirkan kebijakan AWS IoT alias perannya alih-alih membuat yang baru. Lakukan hal-hal berikut:

   a. (Opsional) Buat file yang berisi dokumen AWS IoT kebijakan yang diperlukan alias peran.

      Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

      ```
      nano greengrass-v2-iot-role-alias-policy.json
      ```

      Salin JSON berikut ke dalam file.

- Ganti ARN sumber daya dengan ARN alias peran Anda.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:AssumeRoleWithCertificate",
      "Resource": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
    }
  ]
}
```

b. Buat AWS IoT kebijakan dari dokumen kebijakan.

- Ganti *GreengrassCoreTokenExchangeRoleAliasKebijakan* dengan nama AWS IoT kebijakan yang akan dibuat.

```
aws iot create-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --policy-document file://greengrass-v2-iot-role-alias-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "policyName": "GreengrassCoreTokenExchangeRoleAliasPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassCoreTokenExchangeRoleAliasPolicy",
  "policyDocument": "{
    \\"Version\\":\\"2012-10-17\\",
    \\"Statement\\": [
      {
        \\"Effect\\": \\"Allow\\",
        \\"Action\\": \\"iot:AssumeRoleWithCertificate\\",
        \\"Resource\\": \\"arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias\\"
      }
    ]
  }",
  "policyVersionId": "1"
```

```
}
```

c. Lampirkan AWS IoT kebijakan ke sertifikat AWS IoT benda itu.

- Ganti *GreengrassCoreTokenExchangeRoleAliasKebijakan* dengan nama AWS IoT kebijakan alias peran.
- Ganti ARN target dengan ARN sertifikat untuk objek AWS IoT Anda.

```
aws iot attach-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

Unduh sertifikat ke perangkat

Sebelumnya, Anda mengunduh sertifikat perangkat ke komputer pengembangan Anda. Di bagian ini, Anda menyalin sertifikat ke perangkat inti Anda untuk mengatur perangkat dengan sertifikat yang digunakan untuk terhubung AWS IoT. Anda juga mengunduh sertifikat otoritas sertifikat root Amazon (CA). Jika Anda menggunakan HSM, Anda juga mengimpor file sertifikat ke HSM di bagian ini.

- Jika Anda membuat sertifikat benda dan kunci pribadi di AWS IoT layanan sebelumnya, ikuti langkah-langkah untuk mengunduh sertifikat dengan kunci pribadi dan file sertifikat.
- Jika Anda membuat sertifikat benda dari kunci pribadi dalam modul keamanan perangkat keras (HSM) sebelumnya, ikuti langkah-langkah untuk mengunduh sertifikat dengan kunci pribadi dan sertifikat di HSM.

Unduh sertifikat dengan kunci pribadi dan file sertifikat

Untuk mengunduh sertifikat ke perangkat

1. Salin sertifikat AWS IoT benda dari komputer pengembangan Anda ke perangkat. Jika SSH dan SCP diaktifkan pada komputer pengembangan dan perangkat, Anda dapat menggunakan `scp` perintah di komputer pengembangan Anda untuk mentransfer sertifikat. Ganti *device-ip-address dengan alamat* IP perangkat Anda.

```
scp -r greengrass-v2-certs/ device-ip-address:~
```

2. Buat folder akar Greengrass pada perangkat tersebut. Anda nantinya akan menginstal perangkat lunak AWS IoT Greengrass Core ke folder ini.

   Linux or Unix

   - Ganti */greengrass/v2* dengan folder yang akan digunakan.

   ```
   sudo mkdir -p /greengrass/v2
   ```

   Windows Command Prompt

   - Ganti *C:\greengrass\v2* dengan folder yang akan digunakan.

   ```
   mkdir C:\greengrass\v2
   ```

   PowerShell

   - Ganti *C:\greengrass\v2* dengan folder yang akan digunakan.

   ```
   mkdir C:\greengrass\v2
   ```

3. (Hanya Linux) Atur izin induk folder root Greengrass.

   - Ganti */greengrass* dengan induk dari folder akar.

   ```
   sudo chmod 755 /greengrass
   ```

4. Salin sertifikat AWS IoT benda ke folder root Greengrass.

   Linux or Unix

   - Ganti */greengrass/v2* dengan folder root Greengrass.

   ```
   sudo cp -R ~/greengrass-v2-certs/* /greengrass/v2
   ```

Windows Command Prompt

- Ganti *C:\greengrass\v2* dengan folder yang akan digunakan.

```
robocopy %USERPROFILE%\greengrass-v2-certs C:\greengrass\v2 /E
```

PowerShell

- Ganti *C:\greengrass\v2* dengan folder yang akan digunakan.

```
cp -Path ~\greengrass-v2-certs\* -Destination C:\greengrass\v2
```

5. Unduh sertifikat otoritas sertifikat root Amazon (CA). AWS IoT sertifikat dikaitkan dengan sertifikat CA root Amazon secara default.

   Linux or Unix

   ```
   sudo curl -o /greengrass/v2/AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   Windows Command Prompt (CMD)

   ```
   curl -o C:\greengrass\v2\\AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   PowerShell

   ```
   iwr -Uri https://www.amazontrust.com/repository/AmazonRootCA1.pem -OutFile C:
   \greengrass\v2\\AmazonRootCA1.pem
   ```

Unduh sertifikat dengan kunci pribadi dan sertifikat di HSM

> **Note**
>
> Fitur ini tersedia untuk v2.5.3 dan yang lebih baru dari komponen inti Greengrass. AWS IoT Greengrass saat ini tidak mendukung fitur ini di perangkat inti Windows.

Untuk mengunduh sertifikat ke perangkat

1. Salin sertifikat AWS IoT benda dari komputer pengembangan Anda ke perangkat. Jika SSH dan SCP diaktifkan pada komputer pengembangan dan perangkat, Anda dapat menggunakan `scp` perintah di komputer pengembangan Anda untuk mentransfer sertifikat. Ganti *`device-ip-address dengan alamat`* IP perangkat Anda.

   ```
   scp -r greengrass-v2-certs/ device-ip-address:~
   ```

2. Buat folder akar Greengrass pada perangkat tersebut. Anda nantinya akan menginstal perangkat lunak AWS IoT Greengrass Core ke folder ini.

   Linux or Unix

   - Ganti */greengrass/v2* dengan folder yang akan digunakan.

   ```
   sudo mkdir -p /greengrass/v2
   ```

   Windows Command Prompt

   - Ganti *C:\greengrass\v2* dengan folder yang akan digunakan.

   ```
   mkdir C:\greengrass\v2
   ```

   PowerShell

   - Ganti *C:\greengrass\v2* dengan folder yang akan digunakan.

   ```
   mkdir C:\greengrass\v2
   ```

3. (Hanya Linux) Atur izin induk folder root Greengrass.

   - Ganti */greengrass* dengan induk dari folder akar.

   ```
   sudo chmod 755 /greengrass
   ```

4. Impor file sertifikat benda,`~/greengrass-v2-certs/device.pem.crt`, ke HSM. Periksa dokumentasi untuk HSM Anda untuk mempelajari cara mengimpor sertifikat ke dalamnya. Impor sertifikat menggunakan token, ID slot, PIN pengguna, label objek, dan ID objek yang sama (jika HSM Anda menggunakannya) tempat Anda membuat kunci pribadi di HSM sebelumnya.

   > **Note**
   >
   > Jika Anda membuat kunci pribadi sebelumnya tanpa ID objek, dan sertifikat memiliki ID objek, atur ID objek kunci pribadi ke nilai yang sama dengan sertifikat. Periksa dokumentasi untuk HSM Anda untuk mempelajari cara mengatur ID objek untuk objek kunci pribadi.

5. (Opsional) Hapus file sertifikat benda, sehingga hanya ada di HSM.

   ```
   rm ~/greengrass-v2-certs/device.pem.crt
   ```

6. Unduh sertifikat otoritas sertifikat root Amazon (CA). AWS IoT sertifikat dikaitkan dengan sertifikat CA root Amazon secara default.

   Linux or Unix

   ```
   sudo curl -o /greengrass/v2/AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   Windows Command Prompt (CMD)

   ```
   curl -o C:\greengrass\v2\\AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   PowerShell

   ```
   iwr -Uri https://www.amazontrust.com/repository/AmazonRootCA1.pem -OutFile C:
   \greengrass\v2\\AmazonRootCA1.pem
   ```

Mengatur lingkungan perangkat

Ikuti langkah-langkah di bagian ini untuk menyiapkan perangkat Linux atau Windows untuk digunakan sebagai perangkat AWS IoT Greengrass inti Anda.

Siapkan perangkat Linux

Untuk mengatur perangkat Linux untuk AWS IoT Greengrass V2

1. Instal runtime Java, yang dibutuhkan perangkat lunak AWS IoT Greengrass Core untuk dijalankan. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan. Perintah berikut menunjukkan cara menginstal OpenJDK di perangkat Anda.

   - Untuk distribusi berbasis Debian atau berbasis Ubuntu:

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

   - Untuk distribusi berbasis Red Hat:

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

   - Untuk Amazon Linux 2:

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

   - Untuk Amazon Linux 2023:

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

   Ketika instalasi selesai, jalankan perintah berikut untuk memverifikasi bahwa Java berjalan pada perangkat Linux Anda.

   ```
   java -version
   ```

   Perintah mencetak versi Java yang berjalan pada perangkat. Misalnya, pada distribusi berbasis Debian, output mungkin terlihat mirip dengan sampel berikut.

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

2. (Opsional) Buat pengguna dan grup sistem default yang menjalankan komponen pada perangkat. Anda juga dapat memilih untuk membiarkan penginstal perangkat lunak AWS

IoT Greengrass Core membuat pengguna dan grup ini selama instalasi dengan argumen `--component-default-user` installer. Untuk informasi selengkapnya, lihat Argumen penginstal.

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

3. Verifikasi bahwa pengguna yang menjalankan perangkat lunak AWS IoT Greengrass Core (biasanya`root`), memiliki izin untuk menjalankan `sudo` dengan pengguna dan grup apa pun.

   a. Jalankan perintah berikut untuk membuka `/etc/sudoers` file.

   ```
   sudo visudo
   ```

   b. Verifikasi bahwa izin untuk pengguna terlihat seperti contoh berikut.

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

4. (Opsional) Untuk menjalankan fungsi Lambda kontainer, Anda harus mengaktifkan cgroups v1, dan Anda harus mengaktifkan dan memasang memori dan perangkat cgroups. Jika Anda tidak berencana untuk menjalankan fungsi Lambda kontainer, Anda dapat melewati langkah ini.

   Untuk mengaktifkan opsi cgroups ini, boot perangkat dengan parameter kernel Linux berikut.

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

   Untuk informasi tentang melihat dan menyetel parameter kernel untuk perangkat Anda, lihat dokumentasi untuk sistem operasi dan boot loader Anda. Ikuti instruksi untuk mengatur parameter kernel secara permanen.

5. Instal semua dependensi lain yang diperlukan pada perangkat Anda seperti yang ditunjukkan oleh daftar persyaratan di. Persyaratan perangkat

Siapkan perangkat Windows

> **Note**
>
> Fitur ini tersedia untuk v2.5.0 dan yang lebih baru dari komponen inti Greengrass.

Untuk mengatur perangkat Windows untuk AWS IoT Greengrass V2

1. Instal runtime Java, yang dibutuhkan perangkat lunak AWS IoT Greengrass Core untuk dijalankan. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan.
2. Periksa apakah Java tersedia pada variabel sistem PATH, dan tambahkan jika tidak. LocalSystem Akun menjalankan perangkat lunak AWS IoT Greengrass Core, jadi Anda harus menambahkan Java ke variabel sistem PATH alih-alih variabel pengguna PATH untuk pengguna Anda. Lakukan hal-hal berikut:

   a. Tekan tombol Windows untuk membuka menu mulai.

   b. Ketik **`environment variables`** untuk mencari opsi sistem dari menu mulai.

   c. Di hasil pencarian menu mulai, pilih Edit variabel lingkungan sistem untuk membuka jendela Properti sistem.

   d. Pilih variabel Lingkungan... untuk membuka jendela Variabel Lingkungan.

   e. Di bawah Variabel sistem, pilih Path, lalu pilih Edit. Di jendela variabel Edit lingkungan, Anda dapat melihat setiap jalur pada baris terpisah.

   f. Periksa apakah jalur ke `bin` folder instalasi Java ada. Jalannya mungkin terlihat mirip dengan contoh berikut.

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

   g. Jika `bin` folder instalasi Java hilang dari Path, pilih Baru untuk menambahkannya, lalu pilih OK.
3. Buka Windows Command Prompt (`cmd.exe`) sebagai administrator.
4. Buat pengguna default di LocalSystem akun di perangkat Windows. Ganti *`kata sandi`* dengan kata sandi yang aman.

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

   > **Tip**
   >
   > Bergantung pada konfigurasi Windows Anda, kata sandi pengguna mungkin diatur untuk kedaluwarsa pada tanggal di masa mendatang. Untuk memastikan aplikasi Greengrass Anda terus beroperasi, lacak kapan kata sandi kedaluwarsa, dan perbarui sebelum

kedaluwarsa. Anda juga dapat mengatur kata sandi pengguna agar tidak pernah kedaluwarsa.

- Untuk memeriksa kapan pengguna dan kata sandinya kedaluwarsa, jalankan perintah berikut.

```
net user ggc_user | findstr /C:expires
```

- Untuk mengatur kata sandi pengguna agar tidak pernah kedaluwarsa, jalankan perintah berikut.

```
wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
```

- Jika Anda menggunakan Windows 10 atau yang lebih baru di mana wmicperintah tidak digunakan lagi, jalankan perintah berikut. PowerShell

```
Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
 'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
```

5. Unduh dan instal PsExecutilitas dari Microsoft pada perangkat.
6. Gunakan PsExec utilitas untuk menyimpan nama pengguna dan kata sandi untuk pengguna default dalam contoh Credential Manager untuk LocalSystem akun tersebut. Ganti *kata sandi* dengan kata sandi pengguna yang Anda tetapkan sebelumnya.

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

Jika PsExec License Agreementterbuka, pilih Acceptuntuk menyetujui lisensi dan jalankan perintah.

> **Note**
>
> Pada perangkat Windows, LocalSystem akun menjalankan inti Greengrass, dan Anda harus menggunakan utilitas untuk menyimpan PsExec informasi pengguna default di akun. LocalSystem Menggunakan aplikasi Credential Manager menyimpan informasi ini di akun Windows dari pengguna yang saat ini masuk, bukan LocalSystem akun.

Unduh perangkat lunak AWS IoT Greengrass Inti

Anda dapat mengunduh versi terbaru perangkat lunak AWS IoT Greengrass Core dari lokasi berikut:

- https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip

> **Note**
>
> Anda dapat mengunduh versi tertentu dari perangkat lunak AWS IoT Greengrass Core dari lokasi berikut. Ganti *versi* dengan versi yang akan diunduh.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Untuk mengunduh perangkat lunak AWS IoT Greengrass Core

1. Di perangkat inti Anda, unduh perangkat lunak AWS IoT Greengrass Core ke file bernama`greengrass-nucleus-latest.zip`.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

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

2. (Opsional) Untuk memverifikasi tanda tangan perangkat lunak inti Greengrass

> **Note**
>
> Fitur ini tersedia dengan Greengrass nucleus versi 2.9.5 dan yang lebih baru.

a. Gunakan perintah berikut untuk memverifikasi tanda tangan artefak inti Greengrass Anda:

   Linux or Unix

   ```
   jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
   ```

   Windows Command Prompt (CMD)

   Nama file mungkin terlihat berbeda tergantung pada versi JDK yang Anda instal. Ganti *jdk17.0.6_10*dengan versi JDK yang Anda instal.

   ```
   "C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
   verify -certs -verbose greengrass-nucleus-latest.zip
   ```

   PowerShell

   Nama file mungkin terlihat berbeda tergantung pada versi JDK yang Anda instal. Ganti *jdk17.0.6_10*dengan versi JDK yang Anda instal.

   ```
   'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
   verify -certs -verbose greengrass-nucleus-latest.zip
   ```

b. `jarsigner`Pemanggilan menghasilkan output yang menunjukkan hasil verifikasi.

   i. Jika file zip inti Greengrass ditandatangani, output berisi pernyataan berikut:

   ```
   jar verified.
   ```

   ii. Jika file zip inti Greengrass tidak ditandatangani, output berisi pernyataan berikut:

   ```
   jar is unsigned.
   ```

c. Jika Anda memberikan `-certs` opsi Jarsigner bersama dengan `-verify` dan `-verbose` opsi, output juga menyertakan informasi sertifikat penandatangan terperinci.

3. Buka zip perangkat lunak AWS IoT Greengrass Core ke folder di perangkat Anda. Ganti *GreengrassInstaller*dengan folder yang ingin Anda gunakan.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

4. (Opsional) Jalankan perintah berikut untuk melihat versi perangkat lunak AWS IoT Greengrass Core.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

> **Important**
>
> Jika Anda menginstal versi inti Greengrass lebih awal dari v2.4.0, jangan hapus folder ini setelah Anda menginstal perangkat lunak Core. AWS IoT Greengrass Perangkat lunak AWS IoT Greengrass Core menggunakan file dalam folder ini untuk dijalankan.
> Jika Anda mengunduh versi terbaru perangkat lunak, Anda menginstal v2.4.0 atau yang lebih baru, dan Anda dapat menghapus folder ini setelah Anda menginstal perangkat lunak AWS IoT Greengrass Core.

Instal perangkat lunak AWS IoT Greengrass Inti

Jalankan penginstal dengan argumen yang menentukan tindakan berikut:

- Instal dari file konfigurasi sebagian yang menetapkan untuk menggunakan AWS sumber daya dan sertifikat yang Anda buat sebelumnya. Perangkat lunak AWS IoT Greengrass Core menggunakan file konfigurasi yang menentukan konfigurasi setiap komponen Greengrass pada perangkat. Installer membuat file konfigurasi lengkap dari file konfigurasi parsial yang Anda berikan.
- Tentukan untuk menggunakan pengguna `ggc_user` sistem untuk menjalankan komponen perangkat lunak pada perangkat inti. Pada perangkat Linux, perintah ini juga menentukan untuk menggunakan grup `ggc_group` sistem, dan penginstal membuat pengguna dan grup sistem untuk Anda.
- Siapkan perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem yang berjalan saat boot. Pada perangkat Linux, ini membutuhkan sistem init Systemd.

> **Important**
>
> Pada perangkat inti Windows, Anda harus mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Untuk informasi lebih lanjut tentang argumen yang dapat Anda tentukan, lihat Argumen penginstal.

> **Note**
>
> Jika Anda menjalankan AWS IoT Greengrass perangkat dengan memori terbatas, Anda dapat mengontrol jumlah memori yang digunakan perangkat lunak AWS IoT Greengrass Core. Untuk mengontrol alokasi memori, Anda dapat mengatur pilihan ukuran tumpukan JVM di konfigurasi parameter `jvmOptions` dalam komponen nukleus anda. Untuk informasi selengkapnya, lihat Kontrol alokasi memori dengan opsi JVM.

- Jika Anda membuat sertifikat benda dan kunci pribadi di AWS IoT layanan sebelumnya, ikuti langkah-langkah untuk menginstal perangkat lunak AWS IoT Greengrass inti dengan kunci pribadi dan file sertifikat.
- Jika Anda membuat sertifikat benda dari kunci pribadi dalam modul keamanan perangkat keras (HSM) sebelumnya, ikuti langkah-langkah untuk menginstal perangkat lunak AWS IoT Greengrass Core dengan kunci pribadi dan sertifikat di HSM.

Instal perangkat lunak AWS IoT Greengrass Core dengan kunci pribadi dan file sertifikat

Untuk menginstal perangkat lunak AWS IoT Greengrass Core

1. Periksa versi perangkat lunak AWS IoT Greengrass inti.

   - Ganti *GreengrassInstaller* dengan path ke folder yang berisi perangkat lunak.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Gunakan editor teks untuk membuat file konfigurasi bernama `config.yaml` yang akan disediakan ke penginstal.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   ```
   nano GreengrassInstaller/config.yaml
   ```

   Salin konten YAML berikut ke dalam file. File konfigurasi parsial ini menentukan parameter sistem dan parameter inti Greengrass.

   ```
   ---
   system:
     certificateFilePath: "/greengrass/v2/device.pem.crt"
     privateKeyPath: "/greengrass/v2/private.pem.key"
     rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
     rootpath: "/greengrass/v2"
     thingName: "MyGreengrassCore"
   services:
     aws.greengrass.Nucleus:
       componentType: "NUCLEUS"
       version: "2.12.6"
       configuration:
         awsRegion: "us-west-2"
         iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
         iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
         iotCredEndpoint: "device-credentials-prefix.credentials.iot.us-
   west-2.amazonaws.com"
   ```

   Kemudian, lakukan hal berikut:

- Ganti setiap instance */greengrass/v2* dengan folder root Greengrass.
- Ganti *MyGreengrassCore* dengan nama AWS IoT benda itu.
- Ganti *2.12.6* dengan versi perangkat lunak AWS IoT Greengrass Core.
- Ganti *us-west-2* dengan Wilayah AWS tempat Anda membuat sumber daya.
- Ganti *GreengrassCoreTokenExchangeRoleAlias*dengan nama alias peran pertukaran token.
- Ganti `iotDataEndpoint` dengan titik akhir AWS IoT data Anda.
- Ganti `iotCredEndpoint` dengan titik akhir AWS IoT kredensial Anda.

> **Note**
>
> Dalam file konfigurasi ini, Anda dapat menyesuaikan opsi konfigurasi nukleus lain seperti port dan proksi jaringan yang akan digunakan, seperti yang ditunjukkan dalam contoh berikut. Untuk informasi selengkapnya, lihat konfigurasi nukleus Greengrass.

```
---
system:
  certificateFilePath: "/greengrass/v2/device.pem.crt"
  privateKeyPath: "/greengrass/v2/private.pem.key"
  rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
  rootpath: "/greengrass/v2"
  thingName: "MyGreengrassCore"
services:
  aws.greengrass.Nucleus:
    componentType: "NUCLEUS"
    version: "2.12.6"
    configuration:
      awsRegion: "us-west-2"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      iotCredEndpoint: "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
      mqtt:
        port: 443
      greengrassDataPlanePort: 443
      networkProxy:
        noProxyAddresses: "http://192.168.0.1,www.example.com"
        proxy:
```

```
            url: "https://my-proxy-server:1100"
            username: "Mary_Major"
            password: "pass@word1357"
```

3. Jalankan penginstal, dan tentukan `--init-config` untuk menyediakan file konfigurasi.

   - Ganti */greengrass/v2* atau *C:\greengrass\v2 dengan folder* root Greengrass.
   - Ganti setiap instance *GreengrassInstaller*dengan folder tempat Anda membongkar penginstal.

   Linux or Unix

   ```
   sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
     -jar ./GreengrassInstaller/lib/Greengrass.jar \
     --init-config ./GreengrassInstaller/config.yaml \
     --component-default-user ggc_user:ggc_group \
     --setup-system-service true
   ```

   Windows Command Prompt (CMD)

   ```
   java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" ^
     -jar ./GreengrassInstaller/lib/Greengrass.jar ^
     --init-config ./GreengrassInstaller/config.yaml ^
     --component-default-user ggc_user ^
     --setup-system-service true
   ```

   PowerShell

   ```
   java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" `
     -jar ./GreengrassInstaller/lib/Greengrass.jar `
     --init-config ./GreengrassInstaller/config.yaml `
     --component-default-user ggc_user `
     --setup-system-service true
   ```

> **Important**
>
> Pada perangkat inti Windows, Anda harus menentukan `--setup-system-service true` untuk mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Jika Anda menentukan `--setup-system-service true`, penginstal akan mencetak `Successfully set up Nucleus as a system service` jika ia mengatur dan menjalankan perangkat lunak sebagai layanan. Jika tidak, installer tersebut tidak akan menghasilkan pesan apa pun jika ia berhasil menginstal perangkat lunak tersebut.

> **Note**
>
> Anda tidak dapat menggunakan argumen `deploy-dev-tools` untuk men-deploy alat pengembangan lokal ketika Anda menjalankan penginstal tersebut tanpa argumen `--provision true`. Untuk informasi tentang cara men-deploy Greengrass CLI secara langsung pada perangkat Anda, lihat Antarmuka Baris Perintah Greengrass.

4. Verifikasi instalasi dengan melihat file di folder root.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   dir C:\greengrass\v2
   ```

   PowerShell

   ```
   ls C:\greengrass\v2
   ```

   Jika penginstalan berhasil, folder akar berisi beberapa folder, seperti `config`, `packages`, dan `logs`.

Instal perangkat lunak AWS IoT Greengrass Core dengan kunci pribadi dan sertifikat di HSM

> **Note**
>
> Fitur ini tersedia untuk v2.5.3 dan yang lebih baru dari komponen inti Greengrass. AWS IoT Greengrass saat ini tidak mendukung fitur ini di perangkat inti Windows.

Untuk menginstal perangkat lunak AWS IoT Greengrass Core

1. Periksa versi perangkat lunak AWS IoT Greengrass inti.

   - Ganti *GreengrassInstaller* dengan path ke folder yang berisi perangkat lunak.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Untuk mengaktifkan perangkat lunak AWS IoT Greengrass Core menggunakan kunci pribadi dan sertifikat di HSM, instal komponen penyedia PKCS #11 saat Anda menginstal perangkat lunak Core. AWS IoT Greengrass Komponen penyedia PKCS #11 adalah plugin yang dapat Anda konfigurasi selama instalasi. Anda dapat mengunduh versi terbaru komponen penyedia PKCS #11 dari lokasi berikut:

   - https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/aws.greengrass.crypto.Pkcs11Provider-latest.jar

   Unduh plugin penyedia PKCS #11 ke file bernama. `aws.greengrass.crypto.Pkcs11Provider.jar` Ganti *GreengrassInstaller* dengan folder yang ingin Anda gunakan.

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/
   aws.greengrass.crypto.Pkcs11Provider-latest.jar > GreengrassInstaller/
   aws.greengrass.crypto.Pkcs11Provider.jar
   ```

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

3. Gunakan editor teks untuk membuat file konfigurasi bernama `config.yaml` yang akan disediakan ke penginstal.

Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano GreengrassInstaller/config.yaml
```

Salin konten YAML berikut ke dalam file. File konfigurasi sebagian ini menentukan parameter sistem, parameter inti Greengrass, dan parameter penyedia PKCS #11.

```
---
system:
  certificateFilePath: "pkcs11:object=iotdevicekey;type=cert"
  privateKeyPath: "pkcs11:object=iotdevicekey;type=private"
  rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
  rootpath: "/greengrass/v2"
  thingName: "MyGreengrassCore"
services:
  aws.greengrass.Nucleus:
    componentType: "NUCLEUS"
    version: "2.12.6"
    configuration:
      awsRegion: "us-west-2"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
      iotCredEndpoint: "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
  aws.greengrass.crypto.Pkcs11Provider:
    configuration:
      name: "softhsm_pkcs11"
      library: "/usr/local/Cellar/softhsm/2.6.1/lib/softhsm/libsofthsm2.so"
      slot: 1
      userPin: "1234"
```

Kemudian, lakukan hal berikut:

- Ganti setiap instance *iotdevicekey* di URI PKCS #11 dengan label objek tempat Anda membuat kunci pribadi dan mengimpor sertifikat.
- Ganti setiap instance */greengrass/v2* dengan folder root Greengrass.
- Ganti *MyGreengrassCore* dengan nama AWS IoT benda itu.
- Ganti *2.12.6* dengan versi perangkat lunak AWS IoT Greengrass Core.

- Ganti *us-west-2* dengan Wilayah AWS tempat Anda membuat sumber daya.
- Ganti *GreengrassCoreTokenExchangeRoleAlias*dengan nama alias peran pertukaran token.
- Ganti `iotDataEndpoint` dengan titik akhir AWS IoT data Anda.
- Ganti `iotCredEndpoint` dengan titik akhir AWS IoT kredensial Anda.
- Ganti parameter konfigurasi untuk `aws.greengrass.crypto.Pkcs11Provider` komponen dengan nilai untuk konfigurasi HSM pada perangkat inti.

> **Note**
>
> Dalam file konfigurasi ini, Anda dapat menyesuaikan opsi konfigurasi nukleus lain seperti port dan proksi jaringan yang akan digunakan, seperti yang ditunjukkan dalam contoh berikut. Untuk informasi selengkapnya, lihat konfigurasi nukleus Greengrass.

```
---
system:
  certificateFilePath: "pkcs11:object=iotdevicekey;type=cert"
  privateKeyPath: "pkcs11:object=iotdevicekey;type=private"
  rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
  rootpath: "/greengrass/v2"
  thingName: "MyGreengrassCore"
services:
  aws.greengrass.Nucleus:
    componentType: "NUCLEUS"
    version: "2.12.6"
    configuration:
      awsRegion: "us-west-2"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
      iotCredEndpoint: "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
      mqtt:
        port: 443
      greengrassDataPlanePort: 443
      networkProxy:
        noProxyAddresses: "http://192.168.0.1,www.example.com"
        proxy:
          url: "https://my-proxy-server:1100"
          username: "Mary_Major"
```

```
          password: "pass@word1357"
    aws.greengrass.crypto.Pkcs11Provider:
      configuration:
        name: "softhsm_pkcs11"
        library: "/usr/local/Cellar/softhsm/2.6.1/lib/softhsm/libsofthsm2.so"
        slot: 1
        userPin: "1234"
```

4. Jalankan penginstal, dan tentukan `--init-config` untuk menyediakan file konfigurasi.

   - Ganti */greengrass/v2* dengan folder root Greengrass.
   - Ganti setiap instance *GreengrassInstaller*dengan folder tempat Anda membongkar penginstal.

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
  -jar ./GreengrassInstaller/lib/Greengrass.jar \
  --trusted-plugin ./GreengrassInstaller/aws.greengrass.crypto.Pkcs11Provider.jar \
  --init-config ./GreengrassInstaller/config.yaml \
  --component-default-user ggc_user:ggc_group \
  --setup-system-service true
```

> **Important**
>
> Pada perangkat inti Windows, Anda harus menentukan `--setup-system-service true` untuk mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Jika Anda menentukan `--setup-system-service true`, penginstal akan mencetak `Successfully set up Nucleus as a system service` jika ia mengatur dan menjalankan perangkat lunak sebagai layanan. Jika tidak, installer tersebut tidak akan menghasilkan pesan apa pun jika ia berhasil menginstal perangkat lunak tersebut.

> **Note**
>
> Anda tidak dapat menggunakan argumen `deploy-dev-tools` untuk men-deploy alat pengembangan lokal ketika Anda menjalankan penginstal tersebut tanpa argumen `--`

`provision true`. Untuk informasi tentang cara men-deploy Greengrass CLI secara langsung pada perangkat Anda, lihat Antarmuka Baris Perintah Greengrass.

5. Verifikasi instalasi dengan melihat file di folder root.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   dir C:\greengrass\v2
   ```

   PowerShell

   ```
   ls C:\greengrass\v2
   ```

   Jika penginstalan berhasil, folder akar berisi beberapa folder, seperti `config`, `packages`, dan `logs`.

Jika Anda menginstal perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem, penginstal menjalankan perangkat lunak untuk Anda. Jika tidak, Anda harus menjalankan perangkat lunak itu secara manual. Untuk informasi selengkapnya, lihat Jalankan perangkat lunak inti AWS IoT Greengrass.

Untuk informasi selengkapnya tentang cara mengkonfigurasi dan menggunakan perangkat lunak dan AWS IoT Greengrass, lihat berikut ini:

- Konfigurasikan perangkat lunak AWS IoT Greengrass Inti
- Kembangkan AWS IoT Greengrass komponen
- Deploy komponen AWS IoT Greengrass ke perangkat
- Antarmuka Baris Perintah Greengrass

Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan AWS IoT armada

Fitur ini tersedia untuk v2.4.0 dan yang lebih baru dari komponen inti Greengrass.

Dengan penyediaan AWS IoT armada, Anda dapat mengonfigurasi AWS IoT untuk menghasilkan dan mengirimkan sertifikat perangkat X.509 dan kunci pribadi dengan aman ke perangkat Anda saat terhubung untuk pertama kalinya. AWS IoT AWS IoT menyediakan sertifikat klien yang ditandatangani oleh otoritas sertifikat Amazon Root (CA). Anda juga dapat mengonfigurasi AWS IoT untuk menentukan grup hal, jenis benda, dan izin untuk perangkat inti Greengrass yang Anda sediakan dengan penyediaan armada. Anda menentukan templat penyediaan untuk menentukan bagaimana AWS IoT ketentuan setiap perangkat. Template penyediaan menentukan hal, kebijakan, dan sumber daya sertifikat yang akan dibuat untuk perangkat saat penyediaan. Untuk informasi selengkapnya, lihat Templat penyediaan di Panduan AWS IoT Core Pengembang.

AWS IoT Greengrass menyediakan plugin penyediaan AWS IoT armada yang dapat Anda gunakan untuk menginstal perangkat lunak AWS IoT Greengrass Core menggunakan AWS sumber daya yang dibuat oleh penyediaan AWS IoT armada. Plugin penyediaan armada menggunakan penyediaan berdasarkan klaim. Perangkat menggunakan sertifikat klaim penyediaan dan kunci pribadi untuk mendapatkan sertifikat perangkat X.509 unik dan kunci pribadi yang dapat digunakan untuk operasi reguler. Anda dapat menyematkan sertifikat klaim dan kunci pribadi di setiap perangkat selama pembuatan, sehingga pelanggan Anda dapat mengaktifkan perangkat nanti ketika setiap perangkat online. Anda dapat menggunakan sertifikat klaim dan kunci pribadi yang sama untuk beberapa perangkat. Untuk informasi selengkapnya, lihat Penyediaan berdasarkan klaim di Panduan AWS IoT Core Pengembang.

> **Note**
>
> Plugin penyediaan armada saat ini tidak mendukung penyimpanan kunci pribadi dan file sertifikat dalam modul keamanan perangkat keras (HSM). Untuk menggunakan HSM, instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan manual.

Untuk menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan AWS IoT armada, Anda harus menyiapkan sumber daya Akun AWS yang AWS IoT digunakan untuk menyediakan perangkat inti Greengrass. Sumber daya ini mencakup templat penyediaan, sertifikat klaim, dan peran IAM pertukaran token. Setelah Anda membuat sumber daya ini, Anda dapat menggunakannya

kembali untuk menyediakan beberapa perangkat inti dalam armada. Untuk informasi selengkapnya, lihat Siapkan penyediaan AWS IoT armada untuk perangkat inti Greengrass.

> **Important**
>
> Sebelum Anda mengunduh perangkat lunak AWS IoT Greengrass Core, periksa apakah perangkat inti Anda memenuhi persyaratan untuk menginstal dan menjalankan perangkat lunak AWS IoT Greengrass Core v2.0.

Topik

Prasyarat

Untuk menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan AWS IoT armada, Anda harus terlebih dahulu mengatur penyediaan AWS IoT armada untuk perangkat inti Greengrass. Setelah Anda menyelesaikan langkah-langkah ini sekali, Anda dapat menggunakan penyediaan armada untuk menginstal perangkat lunak AWS IoT Greengrass Core di sejumlah perangkat.

Ambil titik akhir AWS IoT

Dapatkan AWS IoT titik akhir untuk Anda Akun AWS, dan simpan untuk digunakan nanti. Perangkat Anda menggunakan titik akhir ini untuk tersambung ke AWS IoT. Lakukan hal-hal berikut:

1. Dapatkan titik akhir AWS IoT data untuk Anda Akun AWS.

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

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
}
```

2. Dapatkan titik akhir AWS IoT kredensial untuk Anda. Akun AWS

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

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "endpointAddress": "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
}
```

Unduh sertifikat ke perangkat

Perangkat menggunakan sertifikat klaim dan kunci pribadi untuk mengautentikasi permintaannya untuk menyediakan AWS sumber daya dan memperoleh sertifikat perangkat X.509. Anda dapat menyematkan sertifikat klaim dan kunci pribadi ke dalam perangkat selama pembuatan, atau menyalin sertifikat dan kunci ke perangkat selama instalasi. Di bagian ini, Anda menyalin sertifikat klaim dan kunci pribadi ke perangkat. Anda juga mengunduh sertifikat Amazon Root Certificate Authority (CA) ke perangkat.

> **Important**
>
> Kunci pribadi klaim penyediaan harus diamankan setiap saat, termasuk pada perangkat inti Greengrass. Kami menyarankan Anda menggunakan CloudWatch metrik dan log Amazon untuk memantau indikasi penyalahgunaan, seperti penggunaan sertifikat klaim yang tidak sah ke perangkat penyediaan. Jika Anda mendeteksi penyalahgunaan, nonaktifkan sertifikat klaim penyediaan sehingga tidak dapat digunakan untuk penyediaan perangkat. Untuk informasi selengkapnya, lihat Pemantauan AWS IoT di Panduan AWS IoT Core Pengembang.

Untuk membantu mengelola jumlah perangkat dengan lebih baik, dan perangkat mana, yang mendaftarkan diri di perangkat Anda Akun AWS, Anda dapat menentukan hook pra-penyediaan saat membuat templat penyediaan armada. Hook pra-penyediaan adalah AWS Lambda fungsi yang memvalidasi parameter template yang disediakan perangkat selama pendaftaran. Misalnya, Anda dapat membuat hook pra-penyediaan yang memeriksa ID perangkat terhadap database untuk memverifikasi bahwa perangkat memiliki izin untuk menyediakan. Untuk informasi selengkapnya, lihat Pra-penyediaan kait di Panduan Pengembang.AWS IoT Core

Untuk mengunduh sertifikat klaim ke perangkat

1. Salin sertifikat klaim dan kunci pribadi ke perangkat. Jika SSH dan SCP diaktifkan pada komputer pengembangan dan perangkat, Anda dapat menggunakan `scp` perintah di komputer pengembangan Anda untuk mentransfer sertifikat klaim dan kunci pribadi. Contoh perintah berikut mentransfer file-file ini folder bernama `claim-certs` pada komputer pengembangan Anda ke perangkat. Ganti *`device-ip-address dengan alamat`* IP perangkat Anda.

   ```
   scp -r claim-certs/ device-ip-address:~
   ```

2. Buat folder akar Greengrass pada perangkat tersebut. Anda nantinya akan menginstal perangkat lunak AWS IoT Greengrass Core ke folder ini.

   Linux or Unix

   - Ganti */greengrass/v2* dengan folder yang akan digunakan.

   ```
   sudo mkdir -p /greengrass/v2
   ```

   Windows Command Prompt

   - Ganti *C:\greengrass\v2* dengan folder yang akan digunakan.

   ```
   mkdir C:\greengrass\v2
   ```

PowerShell

- Ganti *C:\greengrass\v2* dengan folder yang akan digunakan.

```
mkdir C:\greengrass\v2
```

3. (Hanya Linux) Atur izin induk folder root Greengrass.

   - Ganti */greengrass* dengan induk dari folder akar.

```
sudo chmod 755 /greengrass
```

4. Pindahkan sertifikat klaim ke folder root Greengrass.

   - Ganti */greengrass/v2* atau *C:\greengrass\v2 dengan folder* root Greengrass.

   Linux or Unix

   ```
   sudo mv ~/claim-certs /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   move %USERPROFILE%\claim-certs C:\greengrass\v2
   ```

   PowerShell

   ```
   mv -Path ~\claim-certs -Destination C:\greengrass\v2
   ```

5. Unduh sertifikat otoritas sertifikat root Amazon (CA). AWS IoT sertifikat dikaitkan dengan sertifikat CA root Amazon secara default.

   Linux or Unix

   ```
   sudo curl -o /greengrass/v2/AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

Windows Command Prompt (CMD)

```
curl -o C:\greengrass\v2\\AmazonRootCA1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem
```

PowerShell

```
iwr -Uri https://www.amazontrust.com/repository/AmazonRootCA1.pem -OutFile C:
\greengrass\v2\\AmazonRootCA1.pem
```

Mengatur lingkungan perangkat

Ikuti langkah-langkah di bagian ini untuk menyiapkan perangkat Linux atau Windows untuk digunakan sebagai perangkat AWS IoT Greengrass inti Anda.

Siapkan perangkat Linux

Untuk mengatur perangkat Linux untuk AWS IoT Greengrass V2

1. Instal runtime Java, yang dibutuhkan perangkat lunak AWS IoT Greengrass Core untuk dijalankan. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan. Perintah berikut menunjukkan cara menginstal OpenJDK di perangkat Anda.

   - Untuk distribusi berbasis Debian atau berbasis Ubuntu:

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

   - Untuk distribusi berbasis Red Hat:

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

   - Untuk Amazon Linux 2:

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

   - Untuk Amazon Linux 2023:

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

Ketika instalasi selesai, jalankan perintah berikut untuk memverifikasi bahwa Java berjalan pada perangkat Linux Anda.

```
java -version
```

Perintah mencetak versi Java yang berjalan pada perangkat. Misalnya, pada distribusi berbasis Debian, output mungkin terlihat mirip dengan sampel berikut.

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

2. (Opsional) Buat pengguna dan grup sistem default yang menjalankan komponen pada perangkat. Anda juga dapat memilih untuk membiarkan penginstal perangkat lunak AWS IoT Greengrass Core membuat pengguna dan grup ini selama instalasi dengan argumen `--component-default-user` installer. Untuk informasi selengkapnya, lihat Argumen penginstal.

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

3. Verifikasi bahwa pengguna yang menjalankan perangkat lunak AWS IoT Greengrass Core (biasanya`root`), memiliki izin untuk menjalankan `sudo` dengan pengguna dan grup apa pun.

   a. Jalankan perintah berikut untuk membuka `/etc/sudoers` file.

      ```
      sudo visudo
      ```

   b. Verifikasi bahwa izin untuk pengguna terlihat seperti contoh berikut.

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

4. (Opsional) Untuk menjalankan fungsi Lambda kontainer, Anda harus mengaktifkan cgroups v1, dan Anda harus mengaktifkan dan memasang memori dan perangkat cgroups. Jika Anda tidak berencana untuk menjalankan fungsi Lambda kontainer, Anda dapat melewati langkah ini.

   Untuk mengaktifkan opsi cgroups ini, boot perangkat dengan parameter kernel Linux berikut.

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

Untuk informasi tentang melihat dan menyetel parameter kernel untuk perangkat Anda, lihat dokumentasi untuk sistem operasi dan boot loader Anda. Ikuti instruksi untuk mengatur parameter kernel secara permanen.

5. Instal semua dependensi lain yang diperlukan pada perangkat Anda seperti yang ditunjukkan oleh daftar persyaratan di. Persyaratan perangkat

Siapkan perangkat Windows

> **Note**
>
> Fitur ini tersedia untuk v2.5.0 dan yang lebih baru dari komponen inti Greengrass.

Untuk mengatur perangkat Windows untuk AWS IoT Greengrass V2

1. Instal runtime Java, yang dibutuhkan perangkat lunak AWS IoT Greengrass Core untuk dijalankan. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan.
2. Periksa apakah Java tersedia pada variabel sistem PATH, dan tambahkan jika tidak. LocalSystem Akun menjalankan perangkat lunak AWS IoT Greengrass Core, jadi Anda harus menambahkan Java ke variabel sistem PATH alih-alih variabel pengguna PATH untuk pengguna Anda. Lakukan hal-hal berikut:

   a. Tekan tombol Windows untuk membuka menu mulai.

   b. Ketik **`environment variables`** untuk mencari opsi sistem dari menu mulai.

   c. Di hasil pencarian menu mulai, pilih Edit variabel lingkungan sistem untuk membuka jendela Properti sistem.

   d. Pilih variabel Lingkungan... untuk membuka jendela Variabel Lingkungan.

   e. Di bawah Variabel sistem, pilih Path, lalu pilih Edit. Di jendela variabel Edit lingkungan, Anda dapat melihat setiap jalur pada baris terpisah.

   f. Periksa apakah jalur ke `bin` folder instalasi Java ada. Jalannya mungkin terlihat mirip dengan contoh berikut.

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

g. Jika `bin` folder instalasi Java hilang dari Path, pilih Baru untuk menambahkannya, lalu pilih OK.

3. Buka Windows Command Prompt (`cmd.exe`) sebagai administrator.
4. Buat pengguna default di LocalSystem akun di perangkat Windows. Ganti *`kata sandi`* dengan kata sandi yang aman.

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

> **Tip**
>
> Bergantung pada konfigurasi Windows Anda, kata sandi pengguna mungkin diatur untuk kedaluwarsa pada tanggal di masa mendatang. Untuk memastikan aplikasi Greengrass Anda terus beroperasi, lacak kapan kata sandi kedaluwarsa, dan perbarui sebelum kedaluwarsa. Anda juga dapat mengatur kata sandi pengguna agar tidak pernah kedaluwarsa.
>
> - Untuk memeriksa kapan pengguna dan kata sandinya kedaluwarsa, jalankan perintah berikut.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Untuk mengatur kata sandi pengguna agar tidak pernah kedaluwarsa, jalankan perintah berikut.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Jika Anda menggunakan Windows 10 atau yang lebih baru di mana wmicperintah tidak digunakan lagi, jalankan perintah berikut. PowerShell
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

5. Unduh dan instal PsExecutilitas dari Microsoft pada perangkat.
6. Gunakan PsExec utilitas untuk menyimpan nama pengguna dan kata sandi untuk pengguna default dalam contoh Credential Manager untuk LocalSystem akun tersebut. Ganti *`kata sandi`* dengan kata sandi pengguna yang Anda tetapkan sebelumnya.

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

Jika PsExec License Agreementterbuka, pilih Acceptuntuk menyetujui lisensi dan jalankan perintah.

> **Note**
>
> Pada perangkat Windows, LocalSystem akun menjalankan inti Greengrass, dan Anda harus menggunakan utilitas untuk menyimpan PsExec informasi pengguna default di akun. LocalSystem Menggunakan aplikasi Credential Manager menyimpan informasi ini di akun Windows dari pengguna yang saat ini masuk, bukan LocalSystem akun.

Unduh perangkat lunak AWS IoT Greengrass Core

Anda dapat mengunduh versi terbaru perangkat lunak AWS IoT Greengrass Core dari lokasi berikut:

- https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip

> **Note**
>
> Anda dapat mengunduh versi tertentu dari perangkat lunak AWS IoT Greengrass Core dari lokasi berikut. Ganti *versi* dengan versi yang akan diunduh.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Untuk mengunduh perangkat lunak AWS IoT Greengrass Core

1. Di perangkat inti Anda, unduh perangkat lunak AWS IoT Greengrass Core ke file bernama`greengrass-nucleus-latest.zip`.

   Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

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

2. (Opsional) Untuk memverifikasi tanda tangan perangkat lunak inti Greengrass

> **Note**
>
> Fitur ini tersedia dengan Greengrass nucleus versi 2.9.5 dan yang lebih baru.

a. Gunakan perintah berikut untuk memverifikasi tanda tangan artefak inti Greengrass Anda:

Linux or Unix

```
jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
```

Windows Command Prompt (CMD)

Nama file mungkin terlihat berbeda tergantung pada versi JDK yang Anda instal. Ganti *jdk17.0.6_10*dengan versi JDK yang Anda instal.

```
"C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
verify -certs -verbose greengrass-nucleus-latest.zip
```

PowerShell

Nama file mungkin terlihat berbeda tergantung pada versi JDK yang Anda instal. Ganti *jdk17.0.6_10*dengan versi JDK yang Anda instal.

```
'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
verify -certs -verbose greengrass-nucleus-latest.zip
```

b. `jarsigner`Pemanggilan menghasilkan output yang menunjukkan hasil verifikasi.

   i. Jika file zip inti Greengrass ditandatangani, output berisi pernyataan berikut:

   ```
   jar verified.
   ```

   ii. Jika file zip inti Greengrass tidak ditandatangani, output berisi pernyataan berikut:

   ```
   jar is unsigned.
   ```

c. Jika Anda memberikan `-certs` opsi Jarsigner bersama dengan `-verify` dan `-verbose` opsi, output juga menyertakan informasi sertifikat penandatangan terperinci.

3. Buka zip perangkat lunak AWS IoT Greengrass Core ke folder di perangkat Anda. Ganti *`GreengrassInstaller`*dengan folder yang ingin Anda gunakan.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

4. (Opsional) Jalankan perintah berikut untuk melihat versi perangkat lunak AWS IoT Greengrass Core.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

> **Important**
>
> Jika Anda menginstal versi inti Greengrass lebih awal dari v2.4.0, jangan hapus folder ini setelah Anda menginstal perangkat lunak Core. AWS IoT Greengrass Perangkat lunak AWS IoT Greengrass Core menggunakan file dalam folder ini untuk dijalankan.
> Jika Anda mengunduh versi terbaru perangkat lunak, Anda menginstal v2.4.0 atau yang lebih baru, dan Anda dapat menghapus folder ini setelah Anda menginstal perangkat lunak AWS IoT Greengrass Core.

Unduh plugin penyediaan AWS IoT armada

Anda dapat mengunduh versi terbaru plugin penyediaan AWS IoT armada dari lokasi berikut:

- https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass - FleetProvisioning ByClaim / fleetprovisioningbyclaim-latest.jar

> **Note**
>
> Anda dapat mengunduh versi tertentu dari plugin penyediaan AWS IoT armada dari lokasi berikut. Ganti *versi* dengan versi yang akan diunduh. Untuk informasi selengkapnya tentang setiap versi plugin penyediaan armada, lihat. AWS IoT changelog plugin penyediaan armada
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass-
> FleetProvisioningByClaim/fleetprovisioningbyclaim-version.jar
> ```

Plugin penyediaan armada adalah open source. Untuk melihat kode sumbernya, lihat plugin penyediaan AWS IoT armada di. GitHub

Untuk mengunduh plugin penyediaan AWS IoT armada

- Di perangkat Anda, unduh plugin penyediaan AWS IoT armada ke file bernama. `aws.greengrass.FleetProvisioningByClaim.jar` Ganti *`GreengrassInstaller`*dengan folder yang ingin Anda gunakan.

Linux or Unix

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass-
FleetProvisioningByClaim/fleetprovisioningbyclaim-latest.jar
 > GreengrassInstaller/aws.greengrass.FleetProvisioningByClaim.jar
```

Windows Command Prompt (CMD)

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass-
FleetProvisioningByClaim/fleetprovisioningbyclaim-latest.jar
 > GreengrassInstaller/aws.greengrass.FleetProvisioningByClaim.jar
```

PowerShell

```
iwr -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass-
FleetProvisioningByClaim/fleetprovisioningbyclaim-latest.jar -
OutFile GreengrassInstaller/aws.greengrass.FleetProvisioningByClaim.jar
```

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

Instal perangkat lunak AWS IoT Greengrass Core

Jalankan penginstal dengan argumen yang menentukan tindakan berikut:

- Instal dari file konfigurasi sebagian yang menetapkan untuk menggunakan plugin penyediaan armada untuk menyediakan sumber daya. AWS Perangkat lunak AWS IoT Greengrass Core menggunakan file konfigurasi yang menentukan konfigurasi setiap komponen Greengrass pada perangkat. Penginstal membuat file konfigurasi lengkap dari file konfigurasi paral yang Anda sediakan dan AWS sumber daya yang dibuat oleh plugin penyediaan armada.
- Tentukan untuk menggunakan pengguna `ggc_user` sistem untuk menjalankan komponen perangkat lunak pada perangkat inti. Pada perangkat Linux, perintah ini juga menentukan untuk menggunakan grup `ggc_group` sistem, dan penginstal membuat pengguna dan grup sistem untuk Anda.
- Siapkan perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem yang berjalan saat boot. Pada perangkat Linux, ini membutuhkan sistem init Systemd.

> **Important**
>
> Pada perangkat inti Windows, Anda harus mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Untuk informasi lebih lanjut tentang argumen yang dapat Anda tentukan, lihat Argumen penginstal.

> **Note**
>
> Jika Anda menjalankan AWS IoT Greengrass perangkat dengan memori terbatas, Anda dapat mengontrol jumlah memori yang digunakan perangkat lunak AWS IoT Greengrass Core. Untuk mengontrol alokasi memori, Anda dapat mengatur pilihan ukuran tumpukan JVM di konfigurasi parameter `jvmOptions` dalam komponen nukleus anda. Untuk informasi selengkapnya, lihat Kontrol alokasi memori dengan opsi JVM.

Untuk menginstal perangkat lunak AWS IoT Greengrass Core

1. Periksa versi perangkat lunak AWS IoT Greengrass inti.

   - Ganti *`GreengrassInstaller`*dengan path ke folder yang berisi perangkat lunak.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Gunakan editor teks untuk membuat file konfigurasi bernama `config.yaml` yang akan disediakan ke penginstal.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   ```
   nano GreengrassInstaller/config.yaml
   ```

   Salin konten YAML berikut ke dalam file. File konfigurasi sebagian ini menentukan parameter untuk plugin penyediaan armada. Untuk informasi selengkapnya tentang opsi yang dapat Anda tentukan, lihatKonfigurasikan AWS IoT plugin penyediaan armada.

Linux or Unix

```
---
services:
  aws.greengrass.Nucleus:
    version: "2.12.6"
  aws.greengrass.FleetProvisioningByClaim:
    configuration:
      rootPath: "/greengrass/v2"
      awsRegion: "us-west-2"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
      iotCredentialEndpoint: "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      provisioningTemplate: "GreengrassFleetProvisioningTemplate"
      claimCertificatePath: "/greengrass/v2/claim-certs/claim.pem.crt"
      claimCertificatePrivateKeyPath: "/greengrass/v2/claim-certs/
claim.private.pem.key"
      rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
      templateParameters:
        ThingName: "MyGreengrassCore"
        ThingGroupName: "MyGreengrassCoreGroup"
```

Windows

```
---
services:
  aws.greengrass.Nucleus:
    version: "2.12.6"
  aws.greengrass.FleetProvisioningByClaim:
    configuration:
      rootPath: "C:\\greengrass\\v2"
      awsRegion: "us-west-2"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
      iotCredentialEndpoint: "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      provisioningTemplate: "GreengrassFleetProvisioningTemplate"
      claimCertificatePath: "C:\\greengrass\\v2\\claim-certs\\claim.pem.crt"
      claimCertificatePrivateKeyPath: "C:\\greengrass\\v2\\claim-certs\
\claim.private.pem.key"
      rootCaPath: "C:\\greengrass\\v2\\AmazonRootCA1.pem"
```

```
      templateParameters:
        ThingName: "MyGreengrassCore"
        ThingGroupName: "MyGreengrassCoreGroup"
```

Kemudian, lakukan hal berikut:

- Ganti *2.12.6* dengan versi perangkat lunak AWS IoT Greengrass Core.
- Ganti setiap instance dari */greengrass/v2* atau *C:\greengrass\v2 dengan folder* root Greengrass.

> **Note**
>
> Pada perangkat Windows, Anda harus menentukan pemisah jalur sebagai garis miring terbalik ganda (\\), seperti. `C:\\greengrass\\v2`

- Ganti *us-west-2* dengan Wilayah tempat Anda membuat AWS template penyediaan dan sumber daya lainnya.
- Ganti `iotDataEndpoint` dengan titik akhir AWS IoT data Anda.
- Ganti `iotCredentialEndpoint` dengan titik akhir AWS IoT kredensial Anda.
- Ganti *GreengrassCoreTokenExchangeRoleAlias*dengan nama alias peran pertukaran token.
- Ganti *GreengrassFleetProvisioningTemplate*dengan nama template penyediaan armada.
- Ganti `claimCertificatePath` dengan jalur ke sertifikat klaim pada perangkat.
- Ganti `claimCertificatePrivateKeyPath` dengan jalur ke kunci pribadi sertifikat klaim pada perangkat.
- Ganti parameter template (`templateParameters`) dengan nilai yang akan digunakan untuk menyediakan perangkat. Contoh ini mengacu pada contoh template yang mendefinisikan `ThingName` dan `ThingGroupName` parameter.

> **Note**
>
> Dalam file konfigurasi ini, Anda dapat menyesuaikan opsi konfigurasi lain seperti port dan proxy jaringan yang akan digunakan, seperti yang ditunjukkan pada contoh berikut. Untuk informasi selengkapnya, lihat konfigurasi nukleus Greengrass.

Linux or Unix

```
---
services:
  aws.greengrass.Nucleus:
    version: "2.12.6"
    configuration:
      mqtt:
        port: 443
      greengrassDataPlanePort: 443
      networkProxy:
        noProxyAddresses: "http://192.168.0.1,www.example.com"
        proxy:
          url: "http://my-proxy-server:1100"
          username: "Mary_Major"
          password: "pass@word1357"
  aws.greengrass.FleetProvisioningByClaim:
    configuration:
      rootPath: "/greengrass/v2"
      awsRegion: "us-west-2"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-
west-2.amazonaws.com"
      iotCredentialEndpoint: "device-credentials-
prefix.credentials.iot.us-west-2.amazonaws.com"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      provisioningTemplate: "GreengrassFleetProvisioningTemplate"
      claimCertificatePath: "/greengrass/v2/claim-certs/claim.pem.crt"
      claimCertificatePrivateKeyPath: "/greengrass/v2/claim-certs/
claim.private.pem.key"
      rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
      templateParameters:
        ThingName: "MyGreengrassCore"
        ThingGroupName: "MyGreengrassCoreGroup"
      mqttPort: 443
      proxyUrl: "http://my-proxy-server:1100"
      proxyUserName: "Mary_Major"
      proxyPassword: "pass@word1357"
```

Windows

```
---
services:
```

```
  aws.greengrass.Nucleus:
    version: "2.12.6"
    configuration:
      mqtt:
        port: 443
      greengrassDataPlanePort: 443
      networkProxy:
        noProxyAddresses: "http://192.168.0.1,www.example.com"
        proxy:
          url: "http://my-proxy-server:1100"
          username: "Mary_Major"
          password: "pass@word1357"
  aws.greengrass.FleetProvisioningByClaim:
    configuration:
      rootPath: "C:\\greengrass\\v2"
      awsRegion: "us-west-2"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-
west-2.amazonaws.com"
      iotCredentialEndpoint: "device-credentials-
prefix.credentials.iot.us-west-2.amazonaws.com"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      provisioningTemplate: "GreengrassFleetProvisioningTemplate"
      claimCertificatePath: "C:\\greengrass\\v2\\claim-certs\
\claim.pem.crt"
      claimCertificatePrivateKeyPath: "C:\\greengrass\\v2\\claim-certs\
\claim.private.pem.key"
      rootCaPath: "C:\\greengrass\\v2\\AmazonRootCA1.pem"
      templateParameters:
        ThingName: "MyGreengrassCore"
        ThingGroupName: "MyGreengrassCoreGroup"
      mqttPort: 443
      proxyUrl: "http://my-proxy-server:1100"
      proxyUserName: "Mary_Major"
      proxyPassword: "pass@word1357"
```

Untuk menggunakan proxy HTTPS, Anda harus menggunakan versi 1.1.0 atau yang lebih baru dari plugin penyediaan armada. Anda juga harus menentukan bagian `rootCaPath` bawah`system`, seperti yang ditunjukkan pada contoh berikut.

Linux or Unix

```
---
```

```
system:
  rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
services:
  ...
```

Windows

```
---
system:
  rootCaPath: "C:\\greengrass\\v2\\AmazonRootCA1.pem"
services:
  ...
```

3. Jalankan pemasang. Tentukan `--trusted-plugin` untuk menyediakan plugin penyediaan armada, dan tentukan `--init-config` untuk menyediakan file konfigurasi.

   - Ganti */greengrass/v2* dengan folder root Greengrass.
   - Ganti setiap instance *GreengrassInstaller*dengan folder tempat Anda membongkar penginstal.

   Linux or Unix

   ```
   sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
     -jar ./GreengrassInstaller/lib/Greengrass.jar \
     --trusted-plugin ./GreengrassInstaller/
   aws.greengrass.FleetProvisioningByClaim.jar \
     --init-config ./GreengrassInstaller/config.yaml \
     --component-default-user ggc_user:ggc_group \
     --setup-system-service true
   ```

   Windows Command Prompt (CMD)

   ```
   java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" ^
     -jar ./GreengrassInstaller/lib/Greengrass.jar ^
     --trusted-plugin ./GreengrassInstaller/
   aws.greengrass.FleetProvisioningByClaim.jar ^
     --init-config ./GreengrassInstaller/config.yaml ^
     --component-default-user ggc_user ^
     --setup-system-service true
   ```

PowerShell

```
java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" `
  -jar ./GreengrassInstaller/lib/Greengrass.jar `
  --trusted-plugin ./GreengrassInstaller/
aws.greengrass.FleetProvisioningByClaim.jar `
  --init-config ./GreengrassInstaller/config.yaml `
  --component-default-user ggc_user `
  --setup-system-service true
```

> **Important**
>
> Pada perangkat inti Windows, Anda harus menentukan `--setup-system-service true` untuk mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Jika Anda menentukan `--setup-system-service true`, penginstal akan mencetak `Successfully set up Nucleus as a system service` jika ia mengatur dan menjalankan perangkat lunak sebagai layanan. Jika tidak, installer tersebut tidak akan menghasilkan pesan apa pun jika ia berhasil menginstal perangkat lunak tersebut.

> **Note**
>
> Anda tidak dapat menggunakan argumen `deploy-dev-tools` untuk men-deploy alat pengembangan lokal ketika Anda menjalankan penginstal tersebut tanpa argumen `--provision true`. Untuk informasi tentang cara men-deploy Greengrass CLI secara langsung pada perangkat Anda, lihat Antarmuka Baris Perintah Greengrass.

4. Verifikasi instalasi dengan melihat file di folder root.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   dir C:\greengrass\v2
   ```

PowerShell

```
ls C:\greengrass\v2
```

Jika penginstalan berhasil, folder akar berisi beberapa folder, seperti `config`, `packages`, dan `logs`.

Jika Anda menginstal perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem, penginstal menjalankan perangkat lunak untuk Anda. Jika tidak, Anda harus menjalankan perangkat lunak itu secara manual. Untuk informasi selengkapnya, lihat Jalankan perangkat lunak inti AWS IoT Greengrass.

Untuk informasi selengkapnya tentang cara mengkonfigurasi dan menggunakan perangkat lunak dan AWS IoT Greengrass, lihat berikut ini:

- Konfigurasikan perangkat lunak AWS IoT Greengrass Inti
- Kembangkan AWS IoT Greengrass komponen
- Deploy komponen AWS IoT Greengrass ke perangkat
- Antarmuka Baris Perintah Greengrass

Siapkan penyediaan AWS IoT armada untuk perangkat inti Greengrass

Untuk menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan armada, Anda harus terlebih dahulu menyiapkan sumber daya berikut di perangkat Anda. Akun AWS Sumber daya ini memungkinkan perangkat untuk mendaftarkan diri AWS IoT dan beroperasi sebagai perangkat inti Greengrass. Ikuti langkah-langkah di bagian ini sekali untuk membuat dan mengonfigurasi sumber daya ini di bagian AndaAkun AWS.

- Peran IAM pertukaran token, yang digunakan perangkat inti untuk mengotorisasi panggilan ke AWS layanan.
- Alias AWS IoT peran yang menunjuk ke peran pertukaran token.
- (Opsional) AWS IoT Kebijakan, yang digunakan perangkat inti untuk mengotorisasi panggilan ke AWS IoT dan AWS IoT Greengrass layanan. AWS IoTKebijakan ini harus mengizinkan `iot:AssumeRoleWithCertificate` izin untuk alias AWS IoT peran yang menunjuk ke peran pertukaran token.

Anda dapat menggunakan satu AWS IoT kebijakan untuk semua perangkat inti di armada Anda, atau Anda dapat mengonfigurasi templat penyediaan armada untuk membuat AWS IoT kebijakan untuk setiap perangkat inti.

- AWS IoTTemplate penyediaan armada. Template ini harus menentukan yang berikut:
  - Sumber daya AWS IoT sesuatu. Anda dapat menentukan daftar grup benda yang ada untuk menyebarkan komponen ke setiap perangkat saat online.
  - Sumber daya AWS IoT kebijakan. Sumber daya ini dapat menentukan salah satu properti berikut:
    - Nama AWS IoT kebijakan yang ada. Jika Anda memilih opsi ini, perangkat inti yang Anda buat dari templat ini menggunakan AWS IoT kebijakan yang sama, dan Anda dapat mengelola izinnya sebagai armada.
    - Dokumen AWS IoT kebijakan. Jika Anda memilih opsi ini, setiap perangkat inti yang Anda buat dari templat ini menggunakan AWS IoT kebijakan unik, dan Anda dapat mengelola izin untuk setiap perangkat inti individual.
  - Sumber daya AWS IoT sertifikat. Sumber daya sertifikat ini harus menggunakan `AWS::IoT::Certificate::Id` parameter untuk melampirkan sertifikat ke perangkat inti. Untuk informasi selengkapnya, lihat ust-in-time Penyediaan J di Panduan AWS IoTPengembang.
- Sertifikat klaim AWS IoT penyediaan dan kunci pribadi untuk templat penyediaan armada. Anda dapat menyematkan sertifikat dan kunci pribadi ini di perangkat selama pembuatan, sehingga perangkat dapat mendaftar dan menyediakan sendiri ketika mereka online.

> **Important**
>
> Kunci pribadi klaim penyediaan harus diamankan setiap saat, termasuk pada perangkat inti Greengrass. Kami menyarankan Anda menggunakan CloudWatch metrik dan log Amazon untuk memantau indikasi penyalahgunaan, seperti penggunaan sertifikat klaim yang tidak sah ke perangkat penyediaan. Jika Anda mendeteksi penyalahgunaan, nonaktifkan sertifikat klaim penyediaan sehingga tidak dapat digunakan untuk penyediaan perangkat. Untuk informasi selengkapnya, lihat Pemantauan AWS IoT di Panduan AWS IoT Core Pengembang.
>
> Untuk membantu mengelola jumlah perangkat dengan lebih baik, dan perangkat mana, yang mendaftarkan diri di perangkat AndaAkun AWS, Anda dapat menentukan hook pra-penyediaan saat membuat templat penyediaan armada. Hook pra-penyediaan adalah AWS Lambda fungsi yang memvalidasi parameter template yang disediakan perangkat selama pendaftaran. Misalnya, Anda dapat membuat hook pra-penyediaan yang memeriksa ID

perangkat terhadap database untuk memverifikasi bahwa perangkat memiliki izin untuk menyediakan. Untuk informasi selengkapnya, lihat Pra-penyediaan kait di Panduan Pengembang. AWS IoT Core

- AWS IoTKebijakan yang Anda lampirkan ke sertifikat klaim penyediaan untuk mengizinkan perangkat mendaftar dan menggunakan templat penyediaan armada.

Topik

- Buat peran pertukaran token
- Buat AWS IoT kebijakan
- Buat template penyediaan armada
- Membuat sertifikat klaim penyediaan dan kunci pribadi

Buat peran pertukaran token

Perangkat inti Greengrass menggunakan peran layanan IAM, yang disebut peran pertukaran token, untuk mengotorisasi panggilan ke layanan AWS. Perangkat menggunakan penyedia AWS IoT kredensil untuk mendapatkan AWS kredensil sementara untuk peran ini, yang memungkinkan perangkat berinteraksi, mengirim log ke Amazon LogAWS IoT, dan mengunduh CloudWatch artefak komponen khusus dari Amazon S3. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Anda menggunakan alias AWS IoT peran untuk mengonfigurasi peran pertukaran token untuk perangkat inti Greengrass. Alias peran memungkinkan Anda mengubah peran pertukaran token untuk suatu perangkat tetapi menjaga konfigurasi perangkat tetap sama. Untuk informasi selengkapnya, lihat Mengotorisasi panggilan langsung ke layanan AWS di Panduan Developer AWS IoT Core.

Pada bagian ini, Anda membuat pertukaran token IAM role dan alias peran AWS IoT yang menunjuk ke peran tersebut. Jika Anda telah menyiapkan perangkat inti Greengrass, Anda dapat menggunakan peran pertukaran token dan alias peran alih-alih membuat yang baru.

Buat peran pertukaran token IAM role

1. Buat peran IAM yang dapat digunakan perangkat Anda sebagai peran pertukaran token. Lakukan hal-hal berikut:

a. Buat file yang berisi dokumen kebijakan kepercayaan yang memerlukan peran pertukaran token.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   ```
   nano device-role-trust-policy.json
   ```

   Salin JSON berikut ke dalam file.

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

b. Buat peran pertukaran token dengan dokumen kebijakan kepercayaan.

   - Ganti *GreenGrassV2 TokenExchangeRole* dengan nama peran IAM yang akan dibuat.

   ```
   aws iam create-role --role-name GreengrassV2TokenExchangeRole --assume-role-
   policy-document file://device-role-trust-policy.json
   ```

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "Role": {
       "Path": "/",
       "RoleName": "GreengrassV2TokenExchangeRole",
       "RoleId": "AROAZ2YMUHYHK5OKM77FB",
       "Arn": "arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole",
   ```

```
    "CreateDate": "2021-02-06T00:13:29+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "credentials.iot.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
```

c. Buat file yang berisi dokumen kebijakan akses yang diperlukan oleh peran pertukaran token.

Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano device-role-access-policy.json
```

Salin JSON berikut ke dalam file.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Kebijakan akses ini tidak mengizinkan akses ke artefak komponen dalam bucket S3. Untuk men-deploy komponen kustom yang menentukan artefak di Amazon S3, Anda harus menambahkan izin untuk peran tersebut untuk memungkinkan perangkat inti Anda untuk mengambil artefak komponen. Untuk informasi selengkapnya, lihat Izinkan akses ke bucket S3 untuk artefak komponen.
> Jika Anda belum memiliki bucket S3 untuk artefak komponen, Anda dapat menambahkan izin ini nanti setelah membuat bucket.

d. Buat kebijakan IAM dari dokumen kebijakan.

- Ganti *GreenGrassV2 TokenExchangeRoleAccess* dengan nama kebijakan IAM yang akan dibuat.

```
aws iam create-policy --policy-name GreengrassV2TokenExchangeRoleAccess --
policy-document file://device-role-access-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "Policy": {
    "PolicyName": "GreengrassV2TokenExchangeRoleAccess",
    "PolicyId": "ANPAZ2YMUHYHACI7C5Z66",
    "Arn": "arn:aws:iam::123456789012:policy/
GreengrassV2TokenExchangeRoleAccess",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2021-02-06T00:37:17+00:00",
    "UpdateDate": "2021-02-06T00:37:17+00:00"
  }
}
```

e. Lampirkan kebijakan IAM untuk peran pertukaran token.

- Ganti *GreenGrassV2 TokenExchangeRole* dengan nama peran IAM.

- Ganti ARN peran dengan ARN dari kebijakan IAM yang Anda buat di langkah sebelumnya.

```
aws iam attach-role-policy --role-name GreengrassV2TokenExchangeRole --policy-
arn arn:aws:iam::123456789012:policy/GreengrassV2TokenExchangeRoleAccess
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

2. Buat alias AWS IoT peran yang menunjuk ke peran pertukaran token.

   - Ganti *GreengrassCoreTokenExchangeRoleAlias*dengan nama alias peran yang akan dibuat.
   - Ganti ARN peran dengan ARN dari IAM role yang Anda buat di langkah sebelumnya.

```
aws iot create-role-alias --role-alias GreengrassCoreTokenExchangeRoleAlias --role-
arn arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "roleAlias": "GreengrassCoreTokenExchangeRoleAlias",
  "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
}
```

> **Note**
>
> Untuk membuat alias peran, Anda harus memiliki izin untuk melewati IAM role pertukaran token ke AWS IoT. Jika Anda menerima pesan galat saat mencoba membuat alias peran, periksa apakah AWS pengguna Anda memiliki izin ini. Untuk informasi lebih lanjut, lihat Memberikan izin pengguna untuk meneruskan peran ke layanan AWS di Panduan Pengguna AWS Identity and Access Management.

Buat AWS IoT kebijakan

Setelah Anda mendaftarkan perangkat sebagai AWS IoT sesuatu, perangkat tersebut dapat menggunakan sertifikat digital untuk mengautentikasi. AWS Sertifikat ini mencakup satu atau beberapa AWS IoT kebijakan yang menentukan izin yang dapat digunakan perangkat dengan sertifikat. Kebijakan ini memungkinkan perangkat untuk berkomunikasi dengan AWS IoT danAWS IoT Greengrass.

Dengan penyediaan AWS IoT armada, perangkat terhubung AWS IoT untuk membuat dan mengunduh sertifikat perangkat. Di templat penyediaan armada yang dibuat di bagian berikutnya, Anda dapat menentukan apakah AWS IoT melampirkan AWS IoT kebijakan yang sama ke sertifikat semua perangkat, atau membuat kebijakan baru untuk setiap perangkat.

Di bagian ini, Anda membuat AWS IoT kebijakan yang AWS IoT melekat pada semua sertifikat perangkat. Dengan pendekatan ini, Anda dapat mengelola izin untuk semua perangkat sebagai armada. Jika Anda lebih suka membuat AWS IoT kebijakan baru untuk setiap perangkat, Anda dapat melewati bagian ini, dan merujuk ke kebijakan di dalamnya saat Anda menentukan templat armada Anda.

Untuk membuat AWS IoT kebijakan

- Buat AWS IoT kebijakan yang menentukan AWS IoT izin untuk armada perangkat inti Greengrass Anda. Kebijakan berikut memungkinkan akses ke semua topik MQTT dan operasi Greengrass, sehingga perangkat Anda bekerja dengan aplikasi kustom dan perubahan di masa mendatang yang memerlukan operasi Greengrass baru. Kebijakan ini juga mengizinkan `iot:AssumeRoleWithCertificate` izin, yang memungkinkan perangkat Anda menggunakan peran pertukaran token yang Anda buat di bagian sebelumnya. Anda dapat membatasi kebijakan ini berdasarkan kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2.

  Lakukan hal-hal berikut:

  a. Buat file yang berisi dokumen AWS IoT kebijakan yang dibutuhkan perangkat inti Greengrass.

     Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

     ```
     nano greengrass-v2-iot-policy.json
     ```

Salin JSON berikut ke dalam file.

- Ganti `iot:AssumeRoleWithCertificate` sumber daya dengan ARN alias AWS IoT peran yang Anda buat di bagian sebelumnya.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "iot:Connect",
        "greengrass:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:AssumeRoleWithCertificate",
      "Resource": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
    }
  ]
}
```

b. Buat AWS IoT kebijakan dari dokumen kebijakan.

- Ganti *GreenGrassV2IoT ThingPolicy* dengan nama kebijakan yang akan dibuat.

```
aws iot create-policy --policy-name GreengrassV2IoTThingPolicy --policy-
document file://greengrass-v2-iot-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
```

```
  "policyName": "GreengrassV2IoTThingPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
  "policyDocument": "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
      {
        \"Effect\": \"Allow\",
        \"Action\": [
          \"iot:Publish\",
          \"iot:Subscribe\",
          \"iot:Receive\",
          \"iot:Connect\",
          \"greengrass:*\"
        ],
        \"Resource\": [
          \"*\"
        ]
      },
      {
        \"Effect\": \"Allow\",
        \"Action\": \"iot:AssumeRoleWithCertificate\",
        \"Resource\": \"arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias\"
      }
    ]
  }",
  "policyVersionId": "1"
}
```

Buat template penyediaan armada

AWS IoTTemplat penyediaan armada menentukan cara menyediakan AWS IoT berbagai hal, kebijakan, dan sertifikat. Untuk menyediakan perangkat inti Greengrass dengan plugin penyediaan armada, Anda harus membuat templat yang menentukan hal berikut:

- Sumber daya AWS IoT sesuatu. Anda dapat menentukan daftar grup benda yang ada untuk menyebarkan komponen ke setiap perangkat saat online.
- Sumber daya AWS IoT kebijakan. Sumber daya ini dapat menentukan salah satu properti berikut:

  - Nama AWS IoT kebijakan yang ada. Jika Anda memilih opsi ini, perangkat inti yang Anda buat dari templat ini menggunakan AWS IoT kebijakan yang sama, dan Anda dapat mengelola izinnya sebagai armada.
  - Dokumen AWS IoT kebijakan. Jika Anda memilih opsi ini, setiap perangkat inti yang Anda buat dari templat ini menggunakan AWS IoT kebijakan unik, dan Anda dapat mengelola izin untuk setiap perangkat inti individual.
- Sumber daya AWS IoT sertifikat. Sumber daya sertifikat ini harus menggunakan `AWS::IoT::Certificate::Id` parameter untuk melampirkan sertifikat ke perangkat inti. Untuk informasi selengkapnya, lihat ust-in-time Penyediaan J di Panduan AWS IoTPengembang.

Dalam template, Anda dapat menentukan untuk menambahkan AWS IoT hal ke daftar grup hal yang ada. Ketika perangkat inti terhubung AWS IoT Greengrass untuk pertama kalinya, ia menerima penerapan Greengrass untuk setiap grup hal di mana ia menjadi anggota. Anda dapat menggunakan grup benda untuk menyebarkan perangkat lunak terbaru ke setiap perangkat segera setelah online. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat.

AWS IoTLayanan memerlukan izin untuk membuat dan memperbarui AWS IoT sumber daya di perangkat Anda Akun AWS saat menyediakan perangkat. Untuk memberikan akses AWS IoT layanan, Anda membuat peran IAM dan menyediakannya saat Anda membuat template. AWS IoTmenyediakan kebijakan terkelola AWSIoTThingsRegistration, yang memungkinkan akses ke semua izin yang AWS IoT mungkin digunakan saat menyediakan perangkat. Anda dapat menggunakan kebijakan terkelola ini, atau membuat kebijakan khusus yang mencakup izin dalam kebijakan terkelola untuk kasus penggunaan Anda.

Di bagian ini, Anda membuat peran IAM yang memungkinkan AWS IoT penyediaan sumber daya untuk perangkat, dan Anda membuat templat penyediaan armada yang menggunakan peran IAM tersebut.

Untuk membuat template penyediaan armada

1. Buat peran IAM yang AWS IoT dapat diasumsikan untuk menyediakan sumber daya di AndaAkun AWS. Lakukan hal-hal berikut:

   a. Buat file yang berisi dokumen kebijakan kepercayaan yang memungkinkan AWS IoT untuk mengambil peran.

      Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano aws-iot-trust-policy.json
```

Salin JSON berikut ke dalam file.

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

b. Buat peran IAM dengan dokumen kebijakan kepercayaan.

- Ganti *GreengrassFleetProvisioningRole*dengan nama peran IAM yang akan dibuat.

```
aws iam create-role --role-name GreengrassFleetProvisioningRole --assume-role-
policy-document file://aws-iot-trust-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "Role": {
    "Path": "/",
    "RoleName": "GreengrassFleetProvisioningRole",
    "RoleId": "AROAZ2YMUHYHK5OKM77FB",
    "Arn": "arn:aws:iam::123456789012:role/GreengrassFleetProvisioningRole",
    "CreateDate": "2021-07-26T00:15:12+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
```

```
            "Service": "iot.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}
```

c. Tinjau AWSIoTThingsRegistrationkebijakan, yang memungkinkan akses ke semua izin yang AWS IoT mungkin digunakan saat menyediakan perangkat. Anda dapat menggunakan kebijakan terkelola ini, atau membuat kebijakan khusus yang menentukan izin cakupan bawah untuk kasus penggunaan Anda. Jika Anda memilih untuk membuat kebijakan khusus, lakukan sekarang.

d. Lampirkan kebijakan IAM ke peran penyediaan armada.

- Ganti *GreengrassFleetProvisioningRole*dengan nama peran IAM.
- Jika Anda membuat kebijakan kustom pada langkah sebelumnya, ganti kebijakan ARN dengan ARN dari kebijakan IAM yang akan digunakan.

```
aws iam attach-role-policy --role-name GreengrassFleetProvisioningRole --
policy-arn arn:aws:iam::aws:policy/service-role/AWSIoTThingsRegistration
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

2. (Opsional) Buat hook pra-penyediaan, yang merupakan AWS Lambda fungsi yang memvalidasi parameter template yang disediakan perangkat saat pendaftaran. Anda dapat menggunakan pengait pra-penyediaan untuk mendapatkan kontrol lebih besar atas perangkat mana dan berapa banyak perangkat yang terpasang di dalamnya. Akun AWS Untuk informasi selengkapnya, lihat Pra-penyediaan kait di Panduan Pengembang. AWS IoT Core

3. Buat template penyediaan armada. Lakukan hal-hal berikut:

a. Buat file yang berisi dokumen template penyediaan.

Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano greengrass-fleet-provisioning-template.json
```

Tulis dokumen template penyediaan. Anda dapat mulai dari contoh template penyediaan berikut, yang menentukan untuk membuat AWS IoT sesuatu dengan properti berikut:

- Nama benda adalah nilai yang Anda tentukan dalam parameter `ThingName` template.
- Masalahnya adalah anggota grup benda yang Anda tentukan dalam parameter `ThingGroupName` template. Kelompok benda harus ada di AndaAkun AWS.
- Sertifikat benda itu memiliki AWS IoT kebijakan bernama `GreengrassV2IoTThingPolicy` terlampir padanya.

Untuk informasi selengkapnya, lihat Templat penyediaan di Panduan AWS IoT Core Pengembang.

```
{
  "Parameters": {
    "ThingName": {
      "Type": "String"
    },
    "ThingGroupName": {
      "Type": "String"
    },
    "AWS::IoT::Certificate::Id": {
      "Type": "String"
    }
  },
  "Resources": {
    "MyThing": {
      "OverrideSettings": {
        "AttributePayload": "REPLACE",
        "ThingGroups": "REPLACE",
        "ThingTypeName": "REPLACE"
      },
      "Properties": {
        "AttributePayload": {},
        "ThingGroups": [
          {
            "Ref": "ThingGroupName"
          }
        ],
        "ThingName": {
          "Ref": "ThingName"
```

```
        }
      },
      "Type": "AWS::IoT::Thing"
    },
    "MyPolicy": {
      "Properties": {
        "PolicyName": "GreengrassV2IoTThingPolicy"
      },
      "Type": "AWS::IoT::Policy"
    },
    "MyCertificate": {
      "Properties": {
        "CertificateId": {
          "Ref": "AWS::IoT::Certificate::Id"
        },
        "Status": "Active"
      },
      "Type": "AWS::IoT::Certificate"
    }
  }
}
```

> **Note**
>
> *MyThing*, *MyPolicy*, dan *MyCertificate*merupakan nama arbitrer yang mengidentifikasi setiap spesifikasi sumber daya dalam templat penyediaan armada. AWS IoTtidak menggunakan nama-nama ini dalam sumber daya yang dibuatnya dari template. Anda dapat menggunakan nama-nama ini atau menggantinya dengan nilai yang membantu Anda mengidentifikasi setiap sumber daya dalam template.

b. Buat template penyediaan armada dari dokumen template penyediaan.

- Ganti *GreengrassFleetProvisioningTemplate*dengan nama template yang akan dibuat.
- Ganti deskripsi template dengan deskripsi untuk template Anda.
- Ganti peran penyediaan ARN dengan ARN dari peran yang Anda buat sebelumnya.

Linux or Unix

```
aws iot create-provisioning-template \
```

```
  --template-name GreengrassFleetProvisioningTemplate \
  --description "A provisioning template for Greengrass core devices." \
  --provisioning-role-arn "arn:aws:iam::123456789012:role/
GreengrassFleetProvisioningRole" \
  --template-body file://greengrass-fleet-provisioning-template.json \
  --enabled
```

Windows Command Prompt (CMD)

```
aws iot create-provisioning-template ^
  --template-name GreengrassFleetProvisioningTemplate ^
  --description "A provisioning template for Greengrass core devices." ^
  --provisioning-role-arn "arn:aws:iam::123456789012:role/
GreengrassFleetProvisioningRole" ^
  --template-body file://greengrass-fleet-provisioning-template.json ^
  --enabled
```

PowerShell

```
aws iot create-provisioning-template `
  --template-name GreengrassFleetProvisioningTemplate `
  --description "A provisioning template for Greengrass core devices." `
  --provisioning-role-arn "arn:aws:iam::123456789012:role/
GreengrassFleetProvisioningRole" `
  --template-body file://greengrass-fleet-provisioning-template.json `
  --enabled
```

> **Note**
>
> Jika Anda membuat hook pra-penyediaan, tentukan ARN dari fungsi Lambda hook pra-penyediaan dengan argumen. `--pre-provisioning-hook`
>
> ```
> --pre-provisioning-hook targetArn=arn:aws:lambda:us-
> west-2:123456789012:function:GreengrassPreProvisioningHook
> ```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
```

```
    "templateArn": "arn:aws:iot:us-west-2:123456789012:provisioningtemplate/
GreengrassFleetProvisioningTemplate",
    "templateName": "GreengrassFleetProvisioningTemplate",
    "defaultVersionId": 1
}
```

Membuat sertifikat klaim penyediaan dan kunci pribadi

Sertifikat klaim adalah sertifikat X.509 yang memungkinkan perangkat untuk mendaftar sebagai AWS IoT benda dan mengambil sertifikat perangkat X.509 unik untuk digunakan untuk operasi reguler. Setelah membuat sertifikat klaim, Anda melampirkan AWS IoT kebijakan yang memungkinkan perangkat menggunakannya untuk membuat sertifikat perangkat unik dan penyediaan dengan templat penyediaan armada. Perangkat dengan sertifikat klaim dapat menyediakan hanya menggunakan templat penyediaan yang Anda izinkan dalam kebijakan. AWS IoT

Di bagian ini, Anda membuat sertifikat klaim dan mengonfigurasinya untuk digunakan perangkat dengan templat penyediaan armada yang Anda buat di bagian sebelumnya.

> **Important**
>
> Kunci pribadi klaim penyediaan harus diamankan setiap saat, termasuk pada perangkat inti Greengrass. Kami menyarankan Anda menggunakan CloudWatch metrik dan log Amazon untuk memantau indikasi penyalahgunaan, seperti penggunaan sertifikat klaim yang tidak sah ke perangkat penyediaan. Jika Anda mendeteksi penyalahgunaan, nonaktifkan sertifikat klaim penyediaan sehingga tidak dapat digunakan untuk penyediaan perangkat. Untuk informasi selengkapnya, lihat Pemantauan AWS IoT di Panduan AWS IoT Core Pengembang.
>
> Untuk membantu mengelola jumlah perangkat dengan lebih baik, dan perangkat mana, yang mendaftarkan diri di perangkat AndaAkun AWS, Anda dapat menentukan hook pra-penyediaan saat membuat templat penyediaan armada. Hook pra-penyediaan adalah AWS Lambda fungsi yang memvalidasi parameter template yang disediakan perangkat selama pendaftaran. Misalnya, Anda dapat membuat hook pra-penyediaan yang memeriksa ID perangkat terhadap database untuk memverifikasi bahwa perangkat memiliki izin untuk menyediakan. Untuk informasi selengkapnya, lihat Pra-penyediaan kait di Panduan Pengembang. AWS IoT Core

Untuk membuat sertifikat klaim penyediaan dan kunci pribadi

1. Buat folder tempat Anda mengunduh sertifikat klaim dan kunci pribadi.

   ```
   mkdir claim-certs
   ```

2. Membuat dan menyimpan sertifikat dan kunci pribadi untuk digunakan untuk penyediaan. AWS IoTmenyediakan sertifikat klien yang ditandatangani oleh otoritas sertifikat Amazon Root (CA).

   Linux or Unix

   ```
   aws iot create-keys-and-certificate \
     --certificate-pem-outfile "claim-certs/claim.pem.crt" \
     --public-key-outfile "claim-certs/claim.public.pem.key" \
     --private-key-outfile "claim-certs/claim.private.pem.key" \
     --set-as-active
   ```

   Windows Command Prompt (CMD)

   ```
   aws iot create-keys-and-certificate ^
     --certificate-pem-outfile "claim-certs/claim.pem.crt" ^
     --public-key-outfile "claim-certs/claim.public.pem.key" ^
     --private-key-outfile "claim-certs/claim.private.pem.key" ^
     --set-as-active
   ```

   PowerShell

   ```
   aws iot create-keys-and-certificate `
     --certificate-pem-outfile "claim-certs/claim.pem.crt" `
     --public-key-outfile "claim-certs/claim.public.pem.key" `
     --private-key-outfile "claim-certs/claim.private.pem.key" `
     --set-as-active
   ```

   Tanggapan berisi informasi tentang sertifikat, jika permintaan berhasil. Simpan ARN sertifikat untuk digunakan nanti.

3. Membuat dan melampirkan AWS IoT kebijakan yang memungkinkan perangkat menggunakan sertifikat untuk membuat sertifikat perangkat unik dan penyediaan dengan templat penyediaan armada. Kebijakan berikut memungkinkan akses ke MQTT API penyediaan perangkat. Untuk

informasi selengkapnya, lihat Penyediaan perangkat MQTT API di Panduan Pengembang. AWS IoT Core

Lakukan hal-hal berikut:

a. Buat file yang berisi dokumen AWS IoT kebijakan yang dibutuhkan perangkat inti Greengrass.

Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano greengrass-provisioning-claim-iot-policy.json
```

Salin JSON berikut ke dalam file.

- Ganti setiap instance *wilayah* dengan Wilayah AWS tempat Anda mengatur penyediaan armada.
- Ganti setiap instance *account-id dengan ID* AndaAkun AWS.
- Ganti setiap instance *GreengrassFleetProvisioningTemplate*dengan nama template penyediaan armada yang Anda buat di bagian sebelumnya.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:topic/$aws/certificates/create/*",
        "arn:aws:iot:region:account-id:topic/$aws/provisioning-
templates/GreengrassFleetProvisioningTemplate/provision/*"
      ]
```

```
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:region:account-id:topicfilter/$aws/certificates/create/*",
        "arn:aws:iot:region:account-id:topicfilter/$aws/provisioning-
templates/GreengrassFleetProvisioningTemplate/provision/*"
      ]
    }
  ]
}
```

b. Buat AWS IoT kebijakan dari dokumen kebijakan.

- Ganti *GreengrassProvisioningClaimPolicy*dengan nama kebijakan yang akan dibuat.

```
aws iot create-policy --policy-name GreengrassProvisioningClaimPolicy --policy-
document file://greengrass-provisioning-claim-iot-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "policyName": "GreengrassProvisioningClaimPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassProvisioningClaimPolicy",
  "policyDocument": "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
      {
        \"Effect\": \"Allow\",
        \"Action\": \"iot:Connect\",
        \"Resource\": \"*\"
      },
      {
        \"Effect\": \"Allow\",
        \"Action\": [
          \"iot:Publish\",
          \"iot:Receive\"
        ],
        \"Resource\": [
```

```
          \"arn:aws:iot:region:account-id:topic/$aws/certificates/create/*\",
          \"arn:aws:iot:region:account-id:topic/$aws/provisioning-
templates/GreengrassFleetProvisioningTemplate/provision/*\"
        ]
      },
      {
        \"Effect\": \"Allow\",
        \"Action\": \"iot:Subscribe\",
        \"Resource\": [
          \"arn:aws:iot:region:account-id:topicfilter/$aws/certificates/create/
*\",
          \"arn:aws:iot:region:account-id:topicfilter/$aws/provisioning-
templates/GreengrassFleetProvisioningTemplate/provision/*\"
        ]
      }
    ]
  }",
  "policyVersionId": "1"
}
```

4. Lampirkan AWS IoT kebijakan ke sertifikat klaim penyediaan.

   - Ganti *GreengrassProvisioningClaimPolicy*dengan nama kebijakan yang akan dilampirkan.
   - Ganti ARN target dengan ARN dari sertifikat klaim penyediaan.

   ```
   aws iot attach-policy --policy-name GreengrassProvisioningClaimPolicy --
   target arn:aws:iot:us-west-2:123456789012:cert/
   aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
   ```

   Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

Anda sekarang memiliki sertifikat klaim penyediaan dan kunci pribadi yang dapat digunakan perangkat untuk mendaftar AWS IoT dan menyediakan diri mereka sebagai perangkat inti Greengrass. Anda dapat menyematkan sertifikat klaim dan kunci pribadi di perangkat selama pembuatan, atau menyalin sertifikat dan kunci ke perangkat sebelum Anda menginstal perangkat lunak AWS IoT Greengrass Core. Lihat informasi yang lebih lengkap di Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan AWS IoT armada.

Konfigurasikan AWS IoT plugin penyediaan armada

Plugin penyediaan AWS IoT armada menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan saat menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan armada.

`rootPath`

Jalur ke folder untuk digunakan sebagai root untuk perangkat lunak AWS IoT Greengrass Core.

`awsRegion`

Wilayah AWSYang digunakan plugin penyediaan armada untuk menyediakan sumber daya. AWS

`iotDataEndpoint`

Titik akhir data AWS IoT untuk perangkat Akun AWS.

`iotCredentialEndpoint`

Titik akhir kredensial AWS IoT untuk Akun AWS Anda.

`iotRoleAlias`

Alias AWS IoT peran yang menunjuk ke peran IAM pertukaran token. Penyedia kredensial AWS IoT meneruskan peran ini untuk memungkinkan perangkat inti Greengrass untuk berinteraksi dengan layanan AWS. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

`provisioningTemplate`

Template penyediaan AWS IoT armada untuk digunakan untuk menyediakan AWS sumber daya. Template ini harus menentukan yang berikut:

- Sumber daya AWS IoT sesuatu. Anda dapat menentukan daftar grup benda yang ada untuk menyebarkan komponen ke setiap perangkat saat online.
- Sumber daya AWS IoT kebijakan. Sumber daya ini dapat menentukan salah satu properti berikut:
  - Nama AWS IoT kebijakan yang ada. Jika Anda memilih opsi ini, perangkat inti yang Anda buat dari templat ini menggunakan AWS IoT kebijakan yang sama, dan Anda dapat mengelola izinnya sebagai armada.
  - Dokumen AWS IoT kebijakan. Jika Anda memilih opsi ini, setiap perangkat inti yang Anda buat dari templat ini menggunakan AWS IoT kebijakan unik, dan Anda dapat mengelola izin untuk setiap perangkat inti individual.

- Sumber daya AWS IoT sertifikat. Sumber daya sertifikat ini harus menggunakan `AWS::IoT::Certificate::Id` parameter untuk melampirkan sertifikat ke perangkat inti. Untuk informasi selengkapnya, lihat ust-in-time Penyediaan J di Panduan AWS IoTPengembang.

Untuk informasi selengkapnya, lihat Templat penyediaan di Panduan AWS IoT Core Pengembang.

`claimCertificatePath`

Jalur ke sertifikat klaim penyediaan untuk templat penyediaan yang Anda tentukan. `provisioningTemplate` Untuk informasi selengkapnya, lihat CreateProvisioningClaim di dalam Referensi API AWS IoT Core.

`claimCertificatePrivateKeyPath`

Jalur ke kunci pribadi sertifikat klaim penyediaan untuk templat penyediaan yang Anda tentukan. `provisioningTemplate` Untuk informasi selengkapnya, lihat CreateProvisioningClaim di dalam Referensi API AWS IoT Core.

> **Important**
>
> Kunci pribadi klaim penyediaan harus diamankan setiap saat, termasuk pada perangkat inti Greengrass. Kami menyarankan Anda menggunakan CloudWatch metrik dan log Amazon untuk memantau indikasi penyalahgunaan, seperti penggunaan sertifikat klaim yang tidak sah ke perangkat penyediaan. Jika Anda mendeteksi penyalahgunaan, nonaktifkan sertifikat klaim penyediaan sehingga tidak dapat digunakan untuk penyediaan perangkat. Untuk informasi selengkapnya, lihat Pemantauan AWS IoT di Panduan AWS IoT Core Pengembang.
>
> Untuk membantu mengelola jumlah perangkat dengan lebih baik, dan perangkat mana, yang mendaftarkan diri di perangkat AndaAkun AWS, Anda dapat menentukan hook pra-penyediaan saat membuat templat penyediaan armada. Hook pra-penyediaan adalah AWS Lambda fungsi yang memvalidasi parameter template yang disediakan perangkat selama pendaftaran. Misalnya, Anda dapat membuat hook pra-penyediaan yang memeriksa ID perangkat terhadap database untuk memverifikasi bahwa perangkat memiliki izin untuk menyediakan. Untuk informasi selengkapnya, lihat Pra-penyediaan kait di Panduan Pengembang. AWS IoT Core

`rootCaPath`

Jalur ke sertifikat otoritas sertifikat root Amazon (CA).

`templateParameters`

(Opsional) Peta parameter yang akan disediakan untuk template penyediaan armada. Untuk informasi selengkapnya, lihat bagian Parameter templat penyediaan di Panduan Pengembang. AWS IoT Core

`deviceId`

(Opsional) Pengidentifikasi perangkat yang akan digunakan sebagai ID klien saat plugin penyediaan armada membuat koneksi MQTT ke. AWS IoT

Default: Sebuah UUID acak.

`mqttPort`

(Opsional) Port yang akan digunakan untuk koneksi MQTT.

Default: `8883`

`proxyUrl`

(Opsional) URL server proxy dalam format`scheme://userinfo@host:port`. Untuk menggunakan proxy HTTPS, Anda harus menggunakan versi 1.1.0 atau yang lebih baru dari plugin penyediaan armada.

- `scheme` — Skema, yang harus berupa `http` atau `https`.

  > **Important**
  >
  > Perangkat inti Greengrass harus menjalankan Greengrass nucleus v2.5.0 atau yang lebih baru untuk menggunakan proxy HTTPS.
  > Jika Anda mengonfigurasi proxy HTTPS, Anda harus menambahkan sertifikat CA server proxy ke sertifikat CA root Amazon perangkat inti. Untuk informasi selengkapnya, lihat Aktifkan perangkat inti untuk mempercayai proxy HTTPS.

- `userinfo` - (Opsional) Nama pengguna dan informasi kata sandi. Jika Anda menentukan informasi ini di`url`, perangkat inti Greengrass mengabaikan bidang dan. `username` `password`
- `host` - Nama host atau alamat IP server proksi.
- `port` — (Opsional) Nomor port. Jika Anda tidak menentukan port, maka perangkat inti Greengrass akan menggunakan nilai default berikut:

- `http` – 80
- `https` – 443

`proxyUserName`

(Opsional) Nama pengguna yang mengautentikasi server proxy.

`proxyPassword`

(Opsional) Nama pengguna yang mengautentikasi server proxy.

CSRPath

(Opsional) Jalur ke file permintaan penandatanganan sertifikat (CSR) yang akan digunakan untuk membuat sertifikat perangkat dari CSR. Untuk informasi selengkapnya, lihat Penyediaan berdasarkan klaim di panduan AWS IoT Corepengembang.

csrPrivateKeyJalan

(Opsional, diperlukan jika `csrPath` dideklarasikan) Jalur ke kunci pribadi yang digunakan untuk menghasilkan CSR. Kunci pribadi harus digunakan untuk menghasilkan CSR. Untuk informasi selengkapnya, lihat Penyediaan berdasarkan klaim di panduan AWS IoT Corepengembang.

AWS IoT changelog plugin penyediaan armada

Tabel berikut menjelaskan perubahan dalam setiap versi penyediaan AWS IoT armada oleh plugin klaim ()`aws.greengrass.FleetProvisioningByClaim`.

Versi	Perubahan
1.2.1	Perbaikan bug dan peningkatan • Memperbaiki masalah saat plugin penyediaan armada sedang offline selama startup inti Greengrass. Plugin penyediaan armada sekarang tanpa batas waktu mencoba ulang panggilan koneksi MQTT.
1.2.0	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk penyediaan perangkat melalui permintaan penandatanganan sertifikat dengan jalur kunci pribadi yang dapat dikonfigurasi. • Perbaikan dan perbaikan kecil.

Versi	Perubahan
1.1.0	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk format jalur file tambahan saat Anda mengonfigurasi plugin di perangkat Windows. • Menambahkan dukungan untuk konfigurasi proxy jaringan HTTPS. Lihat informasi yang lebih lengkap di Hubungkan pada port 443 atau melalui proksi jaringan dan Aktifkan perangkat inti untuk mempercayai proxy HTTPS.
1.0.0	Versi awal.

Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya khusus

Fitur ini tersedia untuk v2.4.0 dan yang lebih baru dari komponen inti Greengrass.

Penginstal perangkat lunak AWS IoT Greengrass Core menyediakan antarmuka Java yang dapat Anda terapkan dalam plugin khusus yang menyediakan AWS sumber daya yang diperlukan. Anda dapat mengembangkan plugin penyediaan untuk menggunakan sertifikat klien X.509 kustom atau untuk menjalankan langkah-langkah penyediaan kompleks yang tidak didukung oleh proses instalasi lainnya. Untuk informasi selengkapnya, lihat Membuat sertifikat klien Anda sendiri di Panduan AWS IoT Core Pengembang.

Untuk menjalankan plugin penyediaan khusus saat Anda menginstal perangkat lunak AWS IoT Greengrass Core, Anda membuat file JAR yang Anda berikan kepada penginstal. Penginstal menjalankan plugin, dan plugin mengembalikan konfigurasi penyediaan yang mendefinisikan sumber daya AWS untuk perangkat inti Greengrass. Penginstal menggunakan informasi ini untuk mengkonfigurasi perangkat lunak AWS IoT Greengrass Core pada perangkat. Untuk informasi selengkapnya, lihat Mengembangkan plugin penyediaan kustom.

> **Important**
>
> Sebelum Anda mengunduh perangkat lunak AWS IoT Greengrass Core, periksa apakah perangkat inti Anda memenuhi persyaratan untuk menginstal dan menjalankan perangkat lunak AWS IoT Greengrass Core v2.0.

Topik

Prasyarat

Untuk menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan khusus, Anda harus memiliki yang berikut:

- File JAR untuk plugin penyediaan khusus yang mengimplementasikan file. `DeviceIdentityInterface` Plugin penyediaan khusus harus mengembalikan nilai untuk setiap sistem dan parameter konfigurasi inti. Jika tidak, Anda harus memberikan nilai-nilai tersebut dalam file konfigurasi selama instalasi. Untuk informasi selengkapnya, lihat Mengembangkan plugin penyediaan kustom.

Mengatur lingkungan perangkat

Ikuti langkah-langkah di bagian ini untuk menyiapkan perangkat Linux atau Windows untuk digunakan sebagai perangkat AWS IoT Greengrass inti Anda.

Siapkan perangkat Linux

Untuk mengatur perangkat Linux untuk AWS IoT Greengrass V2

1. Instal runtime Java, yang dibutuhkan perangkat lunak AWS IoT Greengrass Core untuk dijalankan. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan. Perintah berikut menunjukkan cara menginstal OpenJDK di perangkat Anda.

   - Untuk distribusi berbasis Debian atau berbasis Ubuntu:

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

   - Untuk distribusi berbasis Red Hat:

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

- Untuk Amazon Linux 2:

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

- Untuk Amazon Linux 2023:

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

Ketika instalasi selesai, jalankan perintah berikut untuk memverifikasi bahwa Java berjalan pada perangkat Linux Anda.

```
java -version
```

Perintah mencetak versi Java yang berjalan pada perangkat. Misalnya, pada distribusi berbasis Debian, output mungkin terlihat mirip dengan sampel berikut.

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

2. (Opsional) Buat pengguna dan grup sistem default yang menjalankan komponen pada perangkat. Anda juga dapat memilih untuk membiarkan penginstal perangkat lunak AWS IoT Greengrass Core membuat pengguna dan grup ini selama instalasi dengan argumen `--component-default-user` installer. Untuk informasi selengkapnya, lihat Argumen penginstal.

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

3. Verifikasi bahwa pengguna yang menjalankan perangkat lunak AWS IoT Greengrass Core (biasanya`root`), memiliki izin untuk menjalankan `sudo` dengan pengguna dan grup apa pun.

   a. Jalankan perintah berikut untuk membuka `/etc/sudoers` file.

   ```
   sudo visudo
   ```

   b. Verifikasi bahwa izin untuk pengguna terlihat seperti contoh berikut.

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

4. (Opsional) Untuk menjalankan fungsi Lambda kontainer, Anda harus mengaktifkan cgroups v1, dan Anda harus mengaktifkan dan memasang memori dan perangkat cgroups. Jika Anda tidak berencana untuk menjalankan fungsi Lambda kontainer, Anda dapat melewati langkah ini.

   Untuk mengaktifkan opsi cgroups ini, boot perangkat dengan parameter kernel Linux berikut.

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

   Untuk informasi tentang melihat dan menyetel parameter kernel untuk perangkat Anda, lihat dokumentasi untuk sistem operasi dan boot loader Anda. Ikuti instruksi untuk mengatur parameter kernel secara permanen.

5. Instal semua dependensi lain yang diperlukan pada perangkat Anda seperti yang ditunjukkan oleh daftar persyaratan di. Persyaratan perangkat

Siapkan perangkat Windows

> **Note**
>
> Fitur ini tersedia untuk v2.5.0 dan yang lebih baru dari komponen inti Greengrass.

Untuk mengatur perangkat Windows untuk AWS IoT Greengrass V2

1. Instal runtime Java, yang dibutuhkan perangkat lunak AWS IoT Greengrass Core untuk dijalankan. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan.
2. Periksa apakah Java tersedia pada variabel sistem PATH, dan tambahkan jika tidak. LocalSystem Akun menjalankan perangkat lunak AWS IoT Greengrass Core, jadi Anda harus menambahkan Java ke variabel sistem PATH alih-alih variabel pengguna PATH untuk pengguna Anda. Lakukan hal-hal berikut:

   a. Tekan tombol Windows untuk membuka menu mulai.

   b. Ketik **`environment variables`** untuk mencari opsi sistem dari menu mulai.

c. Di hasil pencarian menu mulai, pilih Edit variabel lingkungan sistem untuk membuka jendela Properti sistem.

d. Pilih variabel Lingkungan... untuk membuka jendela Variabel Lingkungan.

e. Di bawah Variabel sistem, pilih Path, lalu pilih Edit. Di jendela variabel Edit lingkungan, Anda dapat melihat setiap jalur pada baris terpisah.

f. Periksa apakah jalur ke `bin` folder instalasi Java ada. Jalannya mungkin terlihat mirip dengan contoh berikut.

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

g. Jika `bin` folder instalasi Java hilang dari Path, pilih Baru untuk menambahkannya, lalu pilih OK.

3. Buka Windows Command Prompt (`cmd.exe`) sebagai administrator.

4. Buat pengguna default di LocalSystem akun di perangkat Windows. Ganti *`kata sandi`* dengan kata sandi yang aman.

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

> **Tip**
>
> Bergantung pada konfigurasi Windows Anda, kata sandi pengguna mungkin diatur untuk kedaluwarsa pada tanggal di masa mendatang. Untuk memastikan aplikasi Greengrass Anda terus beroperasi, lacak kapan kata sandi kedaluwarsa, dan perbarui sebelum kedaluwarsa. Anda juga dapat mengatur kata sandi pengguna agar tidak pernah kedaluwarsa.
>
> - Untuk memeriksa kapan pengguna dan kata sandinya kedaluwarsa, jalankan perintah berikut.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Untuk mengatur kata sandi pengguna agar tidak pernah kedaluwarsa, jalankan perintah berikut.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```

- Jika Anda menggunakan Windows 10 atau yang lebih baru di mana wmicperintah tidak digunakan lagi, jalankan perintah berikut. PowerShell

```
Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
 'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
```

5. Unduh dan instal PsExecutilitas dari Microsoft pada perangkat.
6. Gunakan PsExec utilitas untuk menyimpan nama pengguna dan kata sandi untuk pengguna default dalam contoh Credential Manager untuk LocalSystem akun tersebut. Ganti *kata sandi* dengan kata sandi pengguna yang Anda tetapkan sebelumnya.

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

Jika PsExec License Agreementterbuka, pilih Acceptuntuk menyetujui lisensi dan jalankan perintah.

> **Note**
>
> Pada perangkat Windows, LocalSystem akun menjalankan inti Greengrass, dan Anda harus menggunakan utilitas untuk menyimpan PsExec informasi pengguna default di akun. LocalSystem Menggunakan aplikasi Credential Manager menyimpan informasi ini di akun Windows dari pengguna yang saat ini masuk, bukan LocalSystem akun.

Unduh perangkat lunak AWS IoT Greengrass Inti

Anda dapat mengunduh versi terbaru perangkat lunak AWS IoT Greengrass Core dari lokasi berikut:

- https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip

> **Note**
>
> Anda dapat mengunduh versi tertentu dari perangkat lunak AWS IoT Greengrass Core dari lokasi berikut. Ganti *versi* dengan versi yang akan diunduh.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Untuk mengunduh perangkat lunak AWS IoT Greengrass Core

1. Di perangkat inti Anda, unduh perangkat lunak AWS IoT Greengrass Core ke file bernama`greengrass-nucleus-latest.zip`.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

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

2. (Opsional) Untuk memverifikasi tanda tangan perangkat lunak inti Greengrass

   > **Note**
   >
   > Fitur ini tersedia dengan Greengrass nucleus versi 2.9.5 dan yang lebih baru.

   a. Gunakan perintah berikut untuk memverifikasi tanda tangan artefak inti Greengrass Anda:

      Linux or Unix

      ```
      jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
      ```

Windows Command Prompt (CMD)

Nama file mungkin terlihat berbeda tergantung pada versi JDK yang Anda instal. Ganti *jdk17.0.6_10* dengan versi JDK yang Anda instal.

```
"C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
verify -certs -verbose greengrass-nucleus-latest.zip
```

PowerShell

Nama file mungkin terlihat berbeda tergantung pada versi JDK yang Anda instal. Ganti *jdk17.0.6_10* dengan versi JDK yang Anda instal.

```
'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
verify -certs -verbose greengrass-nucleus-latest.zip
```

b. `jarsigner` Pemanggilan menghasilkan output yang menunjukkan hasil verifikasi.

   i. Jika file zip inti Greengrass ditandatangani, output berisi pernyataan berikut:

   ```
   jar verified.
   ```

   ii. Jika file zip inti Greengrass tidak ditandatangani, output berisi pernyataan berikut:

   ```
   jar is unsigned.
   ```

c. Jika Anda memberikan `-certs` opsi Jarsigner bersama dengan `-verify` dan `-verbose` opsi, output juga menyertakan informasi sertifikat penandatangan terperinci.

3. Buka zip perangkat lunak AWS IoT Greengrass Core ke folder di perangkat Anda. Ganti *GreengrassInstaller* dengan folder yang ingin Anda gunakan.

   Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

4. (Opsional) Jalankan perintah berikut untuk melihat versi perangkat lunak AWS IoT Greengrass Core.

```
java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
```

> **Important**
>
> Jika Anda menginstal versi inti Greengrass lebih awal dari v2.4.0, jangan hapus folder ini setelah Anda menginstal perangkat lunak Core. AWS IoT Greengrass Perangkat lunak AWS IoT Greengrass Core menggunakan file dalam folder ini untuk dijalankan.
> Jika Anda mengunduh versi terbaru perangkat lunak, Anda menginstal v2.4.0 atau yang lebih baru, dan Anda dapat menghapus folder ini setelah Anda menginstal perangkat lunak AWS IoT Greengrass Core.

Instal perangkat lunak AWS IoT Greengrass Core

Jalankan penginstal dengan argumen yang menentukan tindakan berikut:

- Instal dari file konfigurasi sebagian yang menetapkan untuk menggunakan plugin penyediaan kustom Anda untuk menyediakan sumber daya. AWS Perangkat lunak AWS IoT Greengrass Core menggunakan file konfigurasi yang menentukan konfigurasi setiap komponen Greengrass pada perangkat. Penginstal membuat file konfigurasi lengkap dari file konfigurasi paral yang Anda sediakan dan AWS sumber daya yang dibuat oleh plugin penyediaan khusus.
- Tentukan untuk menggunakan pengguna `ggc_user` sistem untuk menjalankan komponen perangkat lunak pada perangkat inti. Pada perangkat Linux, perintah ini juga menentukan untuk

menggunakan grup `ggc_group` sistem, dan penginstal membuat pengguna dan grup sistem untuk Anda.

- Siapkan perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem yang berjalan saat boot. Pada perangkat Linux, ini membutuhkan sistem init Systemd.

> **Important**
>
> Pada perangkat inti Windows, Anda harus mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Untuk informasi lebih lanjut tentang argumen yang dapat Anda tentukan, lihat Argumen penginstal.

> **Note**
>
> Jika Anda menjalankan AWS IoT Greengrass perangkat dengan memori terbatas, Anda dapat mengontrol jumlah memori yang digunakan perangkat lunak AWS IoT Greengrass Core. Untuk mengontrol alokasi memori, Anda dapat mengatur pilihan ukuran tumpukan JVM di konfigurasi parameter `jvmOptions` dalam komponen nukleus anda. Untuk informasi selengkapnya, lihat Kontrol alokasi memori dengan opsi JVM.

Untuk menginstal perangkat lunak AWS IoT Greengrass Core (Linux)

1. Periksa versi perangkat lunak AWS IoT Greengrass inti.

   - Ganti *`GreengrassInstaller`*dengan path ke folder yang berisi perangkat lunak.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Gunakan editor teks untuk membuat file konfigurasi bernama `config.yaml` yang akan disediakan ke penginstal.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   ```
   nano GreengrassInstaller/config.yaml
   ```

Salin konten YAML berikut ke dalam file.

```
---
system:
  rootpath: "/greengrass/v2"
  # The following values are optional. Return them from the provisioning plugin or
 set them here.
  # certificateFilePath: ""
  # privateKeyPath: ""
  # rootCaPath: ""
  # thingName: ""
services:
  aws.greengrass.Nucleus:
    version: "2.12.6"
    configuration:
      # The following values are optional. Return them from the provisioning plugin
 or set them here.
      # awsRegion: ""
      # iotRoleAlias: ""
      # iotDataEndpoint: ""
      # iotCredEndpoint: ""
  com.example.CustomProvisioning:
    configuration:
      # You can specify configuration parameters to provide to your plugin.
      # pluginParameter: ""
```

Kemudian, lakukan hal berikut:

- Ganti *2.12.6* dengan versi perangkat lunak AWS IoT Greengrass Core.
- Ganti setiap instance */greengrass/v2* dengan folder root Greengrass.
- (Opsional) Tentukan nilai konfigurasi sistem dan inti. Anda harus menetapkan nilai-nilai ini jika plugin penyediaan Anda tidak menyediakannya.
- (Opsional) Tentukan parameter konfigurasi yang akan diberikan ke plugin penyediaan Anda.

> **Note**
>
> Dalam file konfigurasi ini, Anda dapat menyesuaikan opsi konfigurasi lainnya, seperti port dan proxy jaringan yang akan digunakan, seperti yang ditunjukkan pada contoh berikut. Untuk informasi selengkapnya, lihat konfigurasi nukleus Greengrass.

```
---
system:
  rootpath: "/greengrass/v2"
  # The following values are optional. Return them from the provisioning
 plugin or set them here.
  # certificateFilePath: ""
  # privateKeyPath: ""
  # rootCaPath: ""
  # thingName: ""
services:
  aws.greengrass.Nucleus:
    version: "2.12.6"
    configuration:
      mqtt:
        port: 443
      greengrassDataPlanePort: 443
      networkProxy:
        noProxyAddresses: "http://192.168.0.1,www.example.com"
        proxy:
          url: "http://my-proxy-server:1100"
          username: "Mary_Major"
          password: "pass@word1357"
      # The following values are optional. Return them from the provisioning
 plugin or set them here.
      # awsRegion: ""
      # iotRoleAlias: ""
      # iotDataEndpoint: ""
      # iotCredEndpoint: ""
  com.example.CustomProvisioning:
    configuration:
      # You can specify configuration parameters to provide to your plugin.
      # pluginParameter: ""
```

3. Jalankan pemasang. Tentukan `--trusted-plugin` untuk menyediakan plugin penyediaan kustom Anda, dan tentukan `--init-config` untuk menyediakan file konfigurasi.

   - Ganti */greengrass/v2* atau *C:\greengrass\v2 dengan folder* root Greengrass.
   - Ganti setiap instance *GreengrassInstaller*dengan folder tempat Anda membongkar penginstal.
   - Ganti jalur ke file JAR plugin penyediaan khusus dengan jalur ke file JAR plugin Anda.

Linux or Unix

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
  -jar ./GreengrassInstaller/lib/Greengrass.jar \
  --trusted-plugin /path/to/com.example.CustomProvisioning.jar \
  --init-config ./GreengrassInstaller/config.yaml \
  --component-default-user ggc_user:ggc_group \
  --setup-system-service true
```

Windows Command Prompt (CMD)

```
java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" ^
  -jar ./GreengrassInstaller/lib/Greengrass.jar ^
  --trusted-plugin /path/to/com.example.CustomProvisioning.jar ^
  --init-config ./GreengrassInstaller/config.yaml ^
  --component-default-user ggc_user ^
  --setup-system-service true
```

PowerShell

```
java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" `
  -jar ./GreengrassInstaller/lib/Greengrass.jar `
  --trusted-plugin /path/to/com.example.CustomProvisioning.jar `
  --init-config ./GreengrassInstaller/config.yaml `
  --component-default-user ggc_user `
  --setup-system-service true
```

Important

Pada perangkat inti Windows, Anda harus menentukan `--setup-system-service true` untuk mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Jika Anda menentukan `--setup-system-service true`, penginstal akan mencetak `Successfully set up Nucleus as a system service` jika ia mengatur dan menjalankan perangkat lunak sebagai layanan. Jika tidak, installer tersebut tidak akan menghasilkan pesan apa pun jika ia berhasil menginstal perangkat lunak tersebut.

> **Note**
>
> Anda tidak dapat menggunakan argumen `deploy-dev-tools` untuk men-deploy alat pengembangan lokal ketika Anda menjalankan penginstal tersebut tanpa argumen `--provision true`. Untuk informasi tentang cara men-deploy Greengrass CLI secara langsung pada perangkat Anda, lihat Antarmuka Baris Perintah Greengrass.

4. Verifikasi instalasi dengan melihat file di folder root.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   dir C:\greengrass\v2
   ```

   PowerShell

   ```
   ls C:\greengrass\v2
   ```

   Jika penginstalan berhasil, folder akar berisi beberapa folder, seperti `config`, `packages`, dan `logs`.

Jika Anda menginstal perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem, penginstal menjalankan perangkat lunak untuk Anda. Jika tidak, Anda harus menjalankan perangkat lunak itu secara manual. Untuk informasi selengkapnya, lihat Jalankan perangkat lunak inti AWS IoT Greengrass.

Untuk informasi selengkapnya tentang cara mengkonfigurasi dan menggunakan perangkat lunak dan AWS IoT Greengrass, lihat berikut ini:

- Konfigurasikan perangkat lunak AWS IoT Greengrass Inti
- Kembangkan AWS IoT Greengrass komponen
- Deploy komponen AWS IoT Greengrass ke perangkat
- Antarmuka Baris Perintah Greengrass

Mengembangkan plugin penyediaan kustom

Untuk mengembangkan plugin penyediaan kustom, membuat kelas Java yang mengimplementasikan`com.aws.greengrass.provisioning.DeviceIdentityInterface`antarmuka. Anda dapat menyertakan file JAR inti Greengrass dalam proyek Anda untuk mengakses antarmuka ini dan kelasnya. Antarmuka ini mendefinisikan metode yang memasukkan konfigurasi plugin dan output konfigurasi penyediaan. Konfigurasi penyediaan mendefinisikan konfigurasi untuk sistem danKomponen inti Greengrass. ParameterAWS IoT GreengrassPenginstal perangkat lunak inti menggunakan konfigurasi penyediaan ini untuk mengkonfigurasiAWS IoT GreengrassPerangkat lunak inti pada perangkat.

Setelah Anda mengembangkan plugin penyediaan kustom, buat sebagai file JAR yang dapat Anda berikan keAWS IoT GreengrassPenginstal perangkat lunak inti untuk menjalankan plugin Anda selama instalasi. Installer menjalankan plugin penyediaan kustom Anda di JVM yang sama yang digunakan installer, sehingga Anda dapat membuat JAR yang hanya berisi kode plugin Anda.

> **Note**
>
> ParameterAWS IoTPlugin penyediaan armadamengaksanakan`DeviceIdentityInterface`untuk menggunakan penyediaan armada selama instalasi. Plugin penyediaan armada adalah open source, sehingga Anda dapat menjelajahi kode sumbernya untuk melihat contoh bagaimana menggunakan antarmuka plugin penyediaan. Untuk informasi selengkapnya, lihatAWS IoTPlugin penyediaan armadadi GitHub.

Topik

- Persyaratan
- Melaksanakan DeviceIdentityInterface antarmuka

Persyaratan

Untuk mengembangkan plugin penyediaan kustom, Anda harus membuat kelas Java yang memenuhi persyaratan berikut:

- Menggunakan`com.aws.greengrass`paket, atau paket dalam`com.aws.greengrass`paket.
- Memiliki konstruktor tanpa argumen.

- MelaksanakanDeviceIdentityInterfaceantarmuka. Untuk informasi selengkapnya, lihat Melaksanakan DeviceIdentityInterface antarmuka.

Melaksanakan DeviceIdentityInterface antarmuka

Untuk menggunakancom.aws.greengrass.provisioning.DeviceIdentityInterfaceantarmuka di plugin kustom Anda, tambahkan inti Greengrass sebagai ketergantungan untuk proyek Anda.

Untuk menggunakan DeviceIdentityInterface dalam proyek plugin penyediaan kustom

- Anda dapat menambahkan file JAR inti Greengrass sebagai perpustakaan, atau menambahkan inti Greengrass sebagai ketergantungan Maven. Lakukan salah satu dari berikut:
  - Untuk menambahkan file JAR inti Greengrass sebagai perpustakaan, unduhAWS IoT Greengrassperangkat lunak inti, yang berisi Greengrass inti JAR. Anda dapat mengunduh versi terbaru perangkat lunak inti AWS IoT Greengrass dari lokasi berikut:
    - https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip

    Anda dapat menemukan file JAR inti Greengrass (`Greengrass.jar`) di`lib`folder dalam file ZIP. Tambahkan file JAR ini ke proyek Anda.
  - Untuk mengkonsumsi nukleus Greengrass dalam proyek Maven, tambahkan ketergantungan pada`nucleus`artefak di`com.aws.greengrass`kelompok. Anda juga harus menambahkan`greengrass-common`repositori, karena inti Greengrass tidak tersedia di Maven Central Repository.

```
<project ...>
    ...
    <repositories>
        <repository>
            <id>greengrass-common</id>
            <name>greengrass common</name>
            <url>https://d2jrmugq4soldf.cloudfront.net/snapshots</url>
        </repository>
    </repositories>
    ...
    <dependencies>
        <dependency>
            <groupId>com.aws.greengrass</groupId>
```

```
            <artifactId>nucleus</artifactId>
            <version>2.5.0-SNAPSHOT</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
</project>
```

Antarmuka DeviceIdentityInterface

Parameter`com.aws.greengrass.provisioning.DeviceIdentityInterface`Antarmuka memiliki bentuk berikut.

> **Note**
>
> Anda juga dapat menjelajahi kelas-kelas ini dipaket com.aws.greengrass.provisioningdariKode sumber inti Greengrassdi GitHub.

```
public interface com.aws.greengrass.provisioning.DeviceIdentityInterface {
    ProvisionConfiguration updateIdentityConfiguration(ProvisionContext context)
            throws RetryableProvisioningException, InterruptedException;

    // Return the name of the plugin.
    String name();
}

com.aws.greengrass.provisioning.ProvisionConfiguration {
    SystemConfiguration systemConfiguration;
    NucleusConfiguration nucleusConfiguration
}

com.aws.greengrass.provisioning.ProvisionConfiguration.SystemConfiguration {
    String certificateFilePath;
    String privateKeyPath;
    String rootCAPath;
    String thingName;
}

com.aws.greengrass.provisioning.ProvisionConfiguration.NucleusConfiguration {
    String awsRegion;
    String iotCredentialsEndpoint;
```

```
    String iotDataEndpoint;
    String iotRoleAlias;
}

com.aws.greengrass.provisioning.ProvisioningContext {
    Map<String, Object> parameterMap;
    String provisioningPolicy;  // The policy is always "PROVISION_IF_NOT_PROVISIONED".
}

com.aws.greengrass.provisioning.exceptions.RetryableProvisioningException {}
```

Setiap nilai konfigurasi di`SystemConfiguration`dan`NucleusConfiguration`diperlukan untuk menginstalAWS IoT GreengrassPerangkat lunak inti, tetapi Anda dapat kembali`null`. Jika plugin penyediaan kustom Anda kembali`null`untuk setiap nilai konfigurasi, Anda harus memberikan nilai dalam sistem atau konfigurasi inti ketika Anda membuat`config.yaml`file untuk memberikan keAWS IoT GreengrassPemasang perangkat lunak inti. Jika plugin penyediaan kustom Anda mengembalikan nilai non-null untuk opsi yang juga Anda tetapkan`config.yaml`, maka installer menggantikan nilai di`config.yaml`dengan nilai yang dikembalikan oleh plugin.

Argumen penginstal

Perangkat lunak inti AWS IoT Greengrass mencakup installer yang menyiapkan perangkat lunak dan menyediakan sumber daya AWS yang diperlukan untuk perangkat inti Greengrass yang akan dijalankan. Installer mencakup argumen berikut yang dapat Anda tentukan untuk mengonfigurasi instalasi:

`-h`, `--help`

(Opsional) Tampilkan informasi bantuan penginstal.

`--version`

(Opsional) Tampilkan versi perangkat lunak inti AWS IoT Greengrass.

`-Droot`

(Opsional) Path ke folder yang akan digunakan sebagai akar untuk perangkat lunak inti AWS IoT Greengrass.

> **Note**
>
> Argumen ini menetapkan properti JVM, sehingga Anda harus menentukannya sebelum `-jar` ketika Anda menjalankan installer tersebut. Sebagai contoh, tentukan `java -Droot="/greengrass/v2" -jar /path/to/Greengrass.jar`.

Default:

- Linux: `~/.greengrass`
- Windows: `%USERPROFILE%/.greengrass`

`-ar`, `--aws-region`

Wilayah AWS yang digunakan oleh perangkat lunak inti AWS IoT Greengrass untuk mengambil atau membuat sumber daya AWS yang diperlukan.

`-p`, `--provision`

(Opsional) Anda dapat mendaftarkan perangkat ini sebagai objek AWS IoT dan menyediakan sumber daya AWS yang dibutuhkan oleh perangkat inti. Jika Anda menentukan `true`, perangkat lunak inti AWS IoT Greengrass akan menyediakan objek AWS IoT, grup objek AWS IoT (opsional), IAM role, dan alias peran AWS IoT.

Default: `false`

`-tn`, `--thing-name`

(Opsional) Nama objek AWS IoT yang Anda daftarkan sebagai perangkat inti ini. Jika objek dengan nama itu tidak ada di Akun AWS, perangkat lunak inti AWS IoT Greengrass akan membuatnya.

> **Note**
>
> Nama objek tidak dapat berisi karakter titik dua (`:`).

Anda harus menentukan `--provision true` untuk menerapkan argumen ini.

Default: `GreengrassV2IotThing_` ditambah UUID acak.

`-tgn`, `--thing-group-name`

(Opsional) Nama grup objek AWS IoT tempat Anda menambahkan objek AWS IoT. Jika deployment menargetkan grup objek ini, perangkat inti ini akan menerima deployment itu ketika terhubung ke AWS IoT Greengrass. Jika grup objek dengan nama ini tidak ada di Akun AWS Anda, perangkat lunak inti AWS IoT Greengrass akan membuatnya.

> **Note**
>
> Nama grup objek tidak dapat berisi karakter titik dua (`:`).

Anda harus menentukan `--provision true` untuk menerapkan argumen ini.

`-tpn`, `--thing-policy-name`

Fitur ini tersedia untuk v2.4.0 dan yang lebih baru dari komponen inti Greengrass.

(Opsional) Nama AWS IoT kebijakan untuk dilampirkan ke sertifikat AWS IoT benda perangkat inti ini. Jika AWS IoT kebijakan dengan nama ini tidak ada di AndaAkun AWS, perangkat lunak AWS IoT Greengrass Core membuatnya.

Perangkat lunak AWS IoT Greengrass Core membuat AWS IoT kebijakan permisif secara default. Anda dapat menjelaskan kebijakan ini, atau membuat kebijakan khusus yang membatasi izin untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2.

Anda harus menentukan `--provision true` untuk menerapkan argumen ini.

Default: `GreengrassV2IoTThingPolicy`

`-trn`, `--tes-role-name`

(Opsional) Nama IAM role yang akan digunakan untuk memperoleh kredensial AWS yang memungkinkan perangkat inti berinteraksi dengan layanan AWS. Jika objek dengan nama itu tidak ada di Akun AWS Anda, perangkat lunak inti AWS IoT Greengrass akan membuatnya dengan kebijakan `GreengrassV2TokenExchangeRoleAccess`. Peran ini tidak memiliki akses ke bucket S3 tempat Anda meng-hosting artefak komponen. Jadi, Anda harus menambahkan izin pada bucket S3 dan objek artefak Anda ketika Anda membuat komponen. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Anda harus menentukan `--provision true` untuk menerapkan argumen ini.

Default: `GreengrassV2TokenExchangeRole`

`-tra`, `--tes-role-alias-name`

(Opsional) Nama alias peran AWS IoT yang menunjuk ke IAM role yang menyediakan kredensial AWS untuk perangkat inti ini. Jika alias peran dengan nama ini tidak ada di Akun AWS, perangkat lunak inti AWS IoT Greengrass akan membuatnya dan mengarahkannya ke IAM role yang Anda tentukan.

Anda harus menentukan `--provision true` untuk menerapkan argumen ini.

Default: `GreengrassV2TokenExchangeRoleAlias`

`-ss`, `--setup-system-service`

(Opsional) Anda dapat mengatur perangkat lunak inti AWS IoT Greengrass sebagai layanan sistem yang berjalan saat perangkat ini booting. Nama layanan sistem adalah `greengrass`. Untuk informasi selengkapnya, lihat Konfigurasikan inti Greengrass sebagai layanan sistem.

Pada sistem operasi Linux, argumen ini mengharuskan sistem init systemd tersedia di perangkat.

> **Important**
>
> Pada perangkat inti Windows, Anda harus mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Default: `false`

`-u`, `--component-default-user`

Nama atau ID pengguna yang digunakan perangkat lunak AWS IoT Greengrass Core untuk menjalankan komponen. Misalnya, Anda dapat menentukan **`ggc_user`**. Nilai ini diperlukan ketika Anda menjalankan installer pada sistem operasi Windows.

Pada sistem operasi Linux, Anda juga dapat menentukan grup secara opsional. Tentukan pengguna dan grup yang dipisahkan dengan titik dua. Misalnya, **`ggc_user:ggc_group`**.

Pertimbangan tambahan berikut berlaku untuk sistem operasi Linux:

- Jika Anda menjalankan sebagai root, pengguna komponen default adalah pengguna yang didefinisikan dalam file konfigurasi. Jika file konfigurasi tidak mendefinisikan pengguna, ini default ke. `ggc_user:ggc_group` Jika `ggc_user` atau `ggc_group` tidak ada, perangkat lunak akan membuatnya.
- Jika Anda berjalan sebagai pengguna non-root, perangkat lunak inti AWS IoT Greengrass akan menggunakan pengguna tersebut untuk menjalankan komponen.
- Jika Anda tidak menentukan grup, perangkat lunak inti AWS IoT Greengrass akan menggunakan grup utama dari pengguna sistem.

Untuk informasi selengkapnya, lihat Konfigurasikan pengguna yang menjalankan komponen.

`-d`, `--deploy-dev-tools`

(Opsional) Anda dapat mengunduh dan men-deploy komponen Greengrass CLI pada perangkat inti ini. Anda dapat menggunakan alat ini untuk mengembangkan dan men-debug komponen pada perangkat inti ini.

> **Important**
>
> Kami menyarankan Anda menggunakan komponen ini hanya di lingkungan pengembangan, bukan lingkungan produksi. Komponen ini menyediakan akses ke informasi dan operasi yang biasanya tidak Anda perlukan di lingkungan produksi. Ikuti prinsip hak istimewa paling sedikit dengan menerapkan komponen ini hanya ke perangkat inti di mana Anda membutuhkannya.

Anda harus menentukan `--provision true` untuk menerapkan argumen ini.

Default: `false`

`-init`, `--init-config`

(Opsional) Path ke file konfigurasi yang akan digunakan untuk menginstal perangkat lunak inti AWS IoT Greengrass. Anda dapat menggunakan opsi ini untuk mengatur perangkat inti baru dengan konfigurasi inti tertentu, misalnya.

> **Important**
>
> File konfigurasi yang Anda tentukan bergabung dengan file konfigurasi yang ada di perangkat inti. Ini termasuk komponen dan konfigurasi komponen pada perangkat inti.

Kami merekomendasikan file konfigurasi hanya mencantumkan konfigurasi yang Anda coba ubah.

`-tp`, `--trusted-plugin`

(Opsional) Jalur ke file JAR untuk dimuat sebagai plugin tepercaya. Gunakan opsi ini untuk menyediakan file JAR plugin penyediaan, seperti menginstal dengan penyediaan armada atau penyediaankhusus, atau untuk menginstal dengan kunci pribadi dan sertifikat dalam modul keamanan perangkat keras.

`-s`, `--start`

(Opsional) Anda dapat memulai perangkat lunak inti AWS IoT Greengrass setelah menginstal dan, secara opsional, menyediakan sumber daya.

Default: `true`

Jalankan perangkat lunak inti AWS IoT Greengrass

Setelah Anda menginstal perangkat lunak inti AWS IoT Greengrass, jalankan perangkat lunak itu untuk menyambungkan perangkat Anda ke AWS IoT Greengrass.

Ketika Anda menginstal perangkat lunak inti AWS IoT Greengrass, Anda dapat menentukan apakah akan menginstalnya sebagai layanan sistem dengan systemd. Jika Anda memilih opsi ini, penginstal akan menjalankan perangkat lunak untuk Anda dan mengonfigurasinya untuk dijalankan saat perangkat Anda melakukan booting.

Important

Pada perangkat inti Windows, Anda harus mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Topik

- Periksa apakah perangkat lunak inti AWS IoT Greengrass berjalan sebagai layanan sistem
- Jalankan perangkat lunak inti AWS IoT Greengrass sebagai layanan sistem

- Jalankan perangkat lunak inti AWS IoT Greengrass tanpa layanan sistem

Periksa apakah perangkat lunak inti AWS IoT Greengrass berjalan sebagai layanan sistem

Ketika Anda menginstal perangkat lunak AWS IoT Greengrass Core, Anda dapat menentukan `--setup-system-service true` argumen untuk menginstal perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem. Perangkat Linux memerlukan sistem init systemd untuk mengatur perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem. Jika Anda menggunakan opsi ini, penginstal akan menjalankan perangkat lunak untuk Anda dan mengonfigurasinya untuk dijalankan saat perangkat Anda melakukan booting. Peginstal mengeluarkan pesan berikut jika berhasil menginstal perangkat lunak inti AWS IoT Greengrass sebagai layanan sistem.

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

Jika Anda sebelumnya telah menginstal perangkat lunak inti AWS IoT Greengrass dan tidak memiliki output penginstal, Anda dapat memeriksa apakah perangkat lunak diinstal sebagai layanan sistem.

Untuk memeriksa apakah perangkat lunak inti AWS IoT Greengrass berjalan sebagai layanan sistem

- Jalankan perintah berikut untuk memeriksa status layanan sistem Greengrass.

  Linux or Unix (systemd)

  ```
  sudo systemctl status greengrass.service
  ```

  Responsnya terlihat serupa dengan contoh berikut jika perangkat lunak inti AWS IoT Greengrass diinstal sebagai layanan sistem dan aktif.

  ```
  # greengrass.service - Greengrass Core
     Loaded: loaded (/etc/systemd/system/greengrass.service; enabled; vendor
   preset: disabled)
     Active: active (running) since Thu 2021-02-11 01:33:44 UTC; 4 days ago
   Main PID: 16107 (sh)
     CGroup: /system.slice/greengrass.service
             ##16107 /bin/sh /greengrass/v2/alts/current/distro/bin/loader
             ##16111 java -Dlog.store=FILE -Droot=/greengrass/v2 -jar /greengrass/
  v2/alts/current/distro/lib/Greengrass...
  ```

Jika `systemctl` atau `greengrass.service` tidak ditemukan, perangkat lunak inti ,AWS IoT Greengrass tidak akan diinstal sebagai layanan sistem. Untuk menjalankan perangkat lunak, lihat Jalankan perangkat lunak inti AWS IoT Greengrass tanpa layanan sistem.

Windows Command Prompt (CMD)

```
sc query greengrass
```

Responsnya terlihat mirip dengan contoh berikut jika perangkat lunak AWS IoT Greengrass Core diinstal sebagai layanan Windows dan aktif.

```
SERVICE_NAME: greengrass
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

PowerShell

```
Get-Service greengrass
```

Responsnya terlihat mirip dengan contoh berikut jika perangkat lunak AWS IoT Greengrass Core diinstal sebagai layanan Windows dan aktif.

```
Status   Name               DisplayName
------   ----               -----------
Running  greengrass         greengrass
```

Jalankan perangkat lunak inti AWS IoT Greengrass sebagai layanan sistem

Jika perangkat lunak inti AWS IoT Greengrass diinstal sebagai layanan sistem, Anda dapat menggunakan manajer layanan sistem untuk memulai, menghentikan, dan mengelola perangkat lunak. Untuk informasi selengkapnya, lihat Konfigurasikan inti Greengrass sebagai layanan sistem.

Untuk menjalankan perangkat lunak AWS IoT Greengrass Core

- Jalankan perintah berikut untuk memulai perangkat lunak inti AWS IoT Greengrass.

  Linux or Unix (systemd)

  ```
  sudo systemctl start greengrass.service
  ```

  Windows Command Prompt (CMD)

  ```
  sc start greengrass
  ```

  PowerShell

  ```
  Start-Service greengrass
  ```

Jalankan perangkat lunak inti AWS IoT Greengrass tanpa layanan sistem

Pada perangkat inti Linux, jika perangkat lunak AWS IoT Greengrass Core tidak diinstal sebagai layanan sistem, Anda dapat menjalankan skrip loader perangkat lunak untuk menjalankan perangkat lunak.

Untuk menjalankan perangkat lunak inti AWS IoT Greengrass tanpa layanan sistem

- Jalankan perintah berikut untuk memulai perangkat lunak inti AWS IoT Greengrass. Jika Anda menjalankan perintah ini di terminal, Anda harus menjaga sesi terminal itu tetap terbuka untuk menjaga perangkat lunak inti AWS IoT Greengrass tetap berjalan.
  - Ganti */greengrass/v2* atau *C:\greengrass\v2 dengan folder* root Greengrass yang Anda gunakan.

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

  Perangkat lunak mencetak pesan berikut jika berhasil meluncurkan.

  ```
  Launched Nucleus successfully.
  ```

Jalankan perangkat lunak AWS IoT Greengrass Core dalam wadah Docker

AWS IoT Greengrass dapat dikonfigurasi untuk dijalankan dalam wadah Docker. Docker adalah sebuah platform yang memungkinkan Anda untuk membangun, menjalankan, menguji, dan men-deploy aplikasi yang didasarkan pada kontainer Linux. Saat Anda menjalankan image AWS IoT Greengrass Docker, Anda dapat memilih apakah akan memberikan AWS kredensyal Anda ke wadah Docker dan mengizinkan penginstal perangkat lunak AWS IoT Greengrass Core untuk secara otomatis menyediakan sumber daya AWS yang diperlukan perangkat inti Greengrass untuk beroperasi. Jika Anda tidak ingin memberikan AWS kredensyal, Anda dapat menyediakan AWS sumber daya secara manual dan menjalankan perangkat lunak AWS IoT Greengrass Core di wadah Docker.

Topik

Platform dan persyaratan yang didukung

Komputer host harus memenuhi persyaratan minimum berikut untuk menginstal dan menjalankan perangkat lunak AWS IoT Greengrass Core dalam wadah Docker:

- Sistem operasi berbasis Linux dengan koneksi internet.
- Mesin Docker versi 18.09 atau yang lebih baru.
- (Opsional) Docker Compose versi 1.22 atau yang lebih baru. Docker Compose diperlukan hanya jika Anda ingin menggunakan Docker Compose CLI untuk menjalankan gambar Docker Anda.

Untuk menjalankan komponen fungsi Lambda dalam kontainer Docker, Anda harus mengonfigurasi kontainer tersebut untuk memenuhi persyaratan tambahan. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.

Jalankan komponen dalam mode proses

AWS IoT Greengrass tidak mendukung menjalankan fungsi Lambda atau komponen yang AWS disediakan di lingkungan runtime yang terisolasi di dalam wadah Docker. AWS IoT Greengrass Anda harus menjalankan komponen ini dalam mode proses tanpa isolasi apapun.

Bila Anda mengonfigurasi komponen fungsi Lambda, atur modus isolasi ke Tidak ada kontainer. Untuk informasi selengkapnya, lihat Jalankan fungsi AWS Lambda.

Saat Anda menerapkan salah satu komponen AWS -provided berikut, perbarui konfigurasi untuk setiap komponen untuk mengatur `containerMode` parameternya. `NoContainer` Untuk informasi selengkapnya tentang pembaruan konfigurasi, lihat Perbarui konfigurasi komponen.

- CloudWatch metrik
- Pembela Perangkat
- Firehose
- Adaptor protokol Modbus-RTU
- Amazon SNS

AWS IoT Greengrass Unduhan perangkat lunak Docker

AWS IoT Greengrass menyediakan Dockerfile untuk membangun image container yang memiliki perangkat lunak AWS IoT Greengrass Core dan dependensi yang diinstal pada image dasar Amazon Linux 2 (x86_64). Anda dapat memodifikasi gambar dasar di Dockerfile untuk berjalan AWS IoT Greengrass pada arsitektur platform yang berbeda.

Unduh paket Dockerfile dari. GitHub

Dockerfile menggunakan versi Greengrass yang lebih lama. Anda harus memperbarui file untuk menggunakan versi Greengrass yang Anda inginkan. Untuk informasi tentang membangun image AWS IoT Greengrass kontainer dari Dockerfile, lihat. Bangun gambar kontainer AWS IoT Greengrass dari Dockerfile

Pilih cara penyediaan sumber daya AWS

Saat Anda menginstal perangkat lunak AWS IoT Greengrass Core dalam wadah Docker, Anda dapat memilih apakah akan secara otomatis menyediakan AWS sumber daya yang diperlukan perangkat inti Greengrass untuk beroperasi, atau menggunakan sumber daya yang Anda sediakan secara manual.

- Penyediaan sumber daya otomatis —Penginstal menyediakan hal, grup AWS IoT AWS IoT benda, peran IAM, dan AWS IoT alias peran saat Anda menjalankan image AWS IoT Greengrass kontainer untuk pertama kalinya. Installer juga dapat men-deploy alat pengembangan lokal ke perangkat inti, sehingga Anda dapat menggunakan perangkat tersebut untuk mengembangkan dan menguji komponen perangkat lunak kustom. Untuk secara otomatis menyediakan sumber daya ini, Anda harus menyediakan kredensial AWS sebagai variabel lingkungan ke gambar Docker.

  Untuk menggunakan penyediaan otomatis, Anda harus mengatur variabel lingkungan Docker `PROVISION=true` dan memasang file kredensial untuk menyediakan kredensial AWS pada kontainer tersebut.
- Penyediaan sumber daya manual —Jika Anda tidak ingin memberikan AWS kredensyal ke wadah, Anda dapat menyediakan AWS sumber daya secara manual sebelum menjalankan image kontainer. AWS IoT Greengrass Anda harus membuat file konfigurasi untuk memberikan informasi tentang sumber daya ini ke penginstal perangkat lunak AWS IoT Greengrass Core dalam wadah Docker.

  Untuk menggunakan penyediaan manual, Anda harus mengatur variabel lingkungan Docker `PROVISION=false`. Penyediaan manual adalah opsi default.

Lihat informasi yang lebih lengkap di Bangun gambar kontainer AWS IoT Greengrass dari Dockerfile.

Bangun gambar kontainer AWS IoT Greengrass dari Dockerfile

AWS menyediakan Dockerfile yang dapat Anda unduh dan gunakan untuk menjalankan perangkat lunak inti AWS IoT Greengrass dalam kontainer Docker. Dockerfile berisi kode sumber untuk membangun citra kontainer AWS IoT Greengrass kustom.

Sebelum Anda membangun sebuah citra kontainer AWS IoT Greengrass, Anda harus mengonfigurasi Dockerfile Anda untuk memilih versi perangkat lunak inti AWS IoT Greengrass yang ingin Anda instal. Anda juga dapat mengonfigurasi variabel lingkungan untuk memilih cara penyediaan sumber daya selama instalasi, dan menyesuaikan opsi instalasi lainnya. Bagian ini

menjelaskan cara mengonfigurasi dan membangun gambar Docker AWS IoT Greengrass dari Dockerfile.

Unduh paket Dockerfile

Anda dapat mengunduh paket AWS IoT Greengrass Dockerfile dari: GitHub

AWS Greengrass Docker Repositori

Setelah Anda men-download paket tersebut, ekstraksi kontennya ke *download-directory/*`aws-greengrass-docker-`*nucleus-version* di komputer Anda. Dockerfile menggunakan versi Greengrass yang lebih lama. Anda harus memperbarui file untuk menggunakan versi Greengrass yang Anda inginkan.

Tentukan versi perangkat lunak inti AWS IoT Greengrass

Gunakan argumen bangun berikut di Dockerfile untuk menentukan versi perangkat lunak inti AWS IoT Greengrass yang ingin Anda gunakan di citra Docker AWS IoT Greengrass. Secara default, Dockerfile menggunakan versi terbaru dari perangkat lunak inti AWS IoT Greengrass.

`GREENGRASS_RELEASE_VERSION`

Versi perangkat lunak inti AWS IoT Greengrass. Secara default, Dockerfile mengunduh versi terbaru yang tersedia dari inti Greengrass. Tetapkan nilai ke versi nukleus yang ingin Anda unduh.

Tetapkan variabel lingkungan

Variabel lingkungan memungkinkan Anda untuk menyesuaikan bagaimana perangkat lunak inti AWS IoT Greengrass diinstal dalam kontainer Docker. Anda dapat mengatur variabel lingkungan untuk gambar Docker AWS IoT Greengrass dalam berbagai cara.

- Untuk menggunakan variabel lingkungan yang sama untuk membuat beberapa gambar, atur variabel lingkungan langsung di Dockerfile.
- Jika Anda menggunakan `docker run` untuk memulai kontainer Anda, lewati variabel lingkungan sebagai argumen dalam perintah, atau atur variabel lingkungan dalam file variabel lingkungan dan kemudian lewati file tersebut sebagai argumen. Untuk informasi selengkapnya tentang pengaturan variabel lingkungan di Docker, lihat variabel lingkungan dalam dokumentasi Docker.

- Jika Anda menggunakan `docker-compose up` untuk memulai kontainer Anda, tetapkan file variabel lingkungan dan kemudian lewati file tersebut sebagai argumen. Untuk informasi selengkapnya tentang pengaturan variabel lingkungan di Compose, lihat Dokumentasi docker.

Anda dapat mengonfigurasi variabel lingkungan berikut untuk citra Docker AWS IoT Greengrass.

> **Note**
>
> Jangan memodifikasi variabel `TINI_KILL_PROCESS_GROUP` dalam Dockerfile. Variabel ini memungkinkan penerusan `SIGTERM` ke semua PID dalam grup PID sehingga perangkat lunak AWS IoT Greengrass Core dapat dimatikan dengan benar ketika wadah Docker dihentikan.

`GGC_ROOT_PATH`

(Opsional) Path ke folder dalam kontainer yang akan digunakan sebagai akar untuk perangkat lunak inti AWS IoT Greengrass.

Default: `/greengrass/v2`

`PROVISION`

(Opsional) Menentukan apakah inti AWS IoT Greengrass menyediakan sumber daya AWS.

- Jika Anda menentukan `true`, perangkat lunak inti AWS IoT Greengrass akan mendaftarkan gambar kontainer tersebut sebagai objek AWS IoT dan menyediakan sumber daya AWS yang diperlukan oleh perangkat inti Greengrass. Perangkat lunak inti AWS IoT Greengrass menyediakan objek AWS IoT, grup objek AWS IoT (opsional), IAM role, dan alias peran AWS IoT. Untuk informasi selengkapnya, lihat Jalankan AWS IoT Greengrass dalam kontainer Docker dengan penyediaan sumber daya otomatis.
- Jika Anda menentukan `false`, maka Anda harus membuat file konfigurasi untuk memberikan ke installer inti AWS IoT Greengrass yang menentukan untuk menggunakan sumber daya AWS dan sertifikat yang Anda buat secara manual. Untuk informasi selengkapnya, lihat Jalankan AWS IoT Greengrass dalam kontainer Docker dengan penyediaan sumber daya manual.

Default: `false`

`AWS_REGION`

(Opsional) Wilayah AWS yang digunakan oleh perangkat lunak inti AWS IoT Greengrass untuk mengambil atau membuat sumber daya AWS yang diperlukan.

Bawaan: `us-east-1`.

`THING_NAME`

(Opsional) Nama objek AWS IoT yang Anda daftarkan sebagai perangkat inti ini. Jika objek dengan nama ini tidak ada di Akun AWS Anda, perangkat lunak inti AWS IoT Greengrass akan membuatnya.

Anda harus menentukan `PROVISION=true` untuk menerapkan argumen ini.

Default: `GreengrassV2IotThing_` ditambah UUID acak.

`THING_GROUP_NAME`

(Opsional) Nama grup objek AWS IoT di mana Anda menambahkan AWS IoT jika deployment menargetkan grup objek ini, hal ini dan perangkat inti lainnya dalam grup tersebut akan menerima deployment tersebut saat tersambung ke AWS IoT Greengrass. Jika grup objek dengan nama ini tidak ada di Akun AWS Anda, perangkat lunak inti AWS IoT Greengrass akan membuatnya.

Anda harus menentukan `PROVISION=true` untuk menerapkan argumen ini.

`TES_ROLE_NAME`

(Opsional) Nama IAM role yang akan digunakan untuk memperoleh kredensial AWS yang memungkinkan perangkat inti Greengrass berinteraksi dengan layanan AWS. Jika objek dengan nama itu tidak ada di Akun AWS Anda, perangkat lunak inti AWS IoT Greengrass akan membuatnya dengan kebijakan `GreengrassV2TokenExchangeRoleAccess`. Peran ini tidak memiliki akses ke bucket S3 tempat Anda meng-hosting artefak komponen. Jadi, Anda harus menambahkan izin pada bucket S3 dan objek artefak Anda ketika Anda membuat komponen. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Default: `GreengrassV2TokenExchangeRole`

`TES_ROLE_ALIAS_NAME`

(Opsional) Nama alias peran AWS IoT yang menunjuk ke IAM role yang menyediakan kredensial AWS untuk perangkat lunak inti Greengrass tersebut. Jika alias peran dengan nama ini tidak ada

di Akun AWS, perangkat lunak inti AWS IoT Greengrass akan membuatnya dan mengarahkannya ke IAM role yang Anda tentukan.

Default: `GreengrassV2TokenExchangeRoleAlias`

`COMPONENT_DEFAULT_USER`

(Opsional) Nama atau ID dari pengguna sistem dan grup yang digunakan oleh perangkat lunak inti AWS IoT Greengrass untuk menjalankan komponen. Tentukan pengguna dan grup, yang dipisahkan dengan titik dua. Grup ini opsional. Misalnya, Anda dapat menentukan **`ggc_user:ggc_group`** atau **`ggc_user`**.

- Jika Anda berjalan sebagai root, hal ini akan menjadi default untuk pengguna dan grup yang ditentukan oleh file konfigurasi. Jika file konfigurasi tersebut tidak menentukan pengguna dan grup, default-nya menjadi `ggc_user:ggc_group`. Jika `ggc_user` atau `ggc_group` tidak ada, perangkat lunak akan membuatnya.
- Jika Anda berjalan sebagai pengguna non-root, perangkat lunak inti AWS IoT Greengrass akan menggunakan pengguna tersebut untuk menjalankan komponen.
- Jika Anda tidak menentukan grup, perangkat lunak inti AWS IoT Greengrass akan menggunakan grup utama dari pengguna sistem.

Untuk informasi selengkapnya, lihat Konfigurasikan pengguna yang menjalankan komponen.

`DEPLOY_DEV_TOOLS`

Mendefinisikan apakah akan mengunduh dan menyebarkan komponen CLI Greengrass dalam gambar kontainer. Anda dapat menggunakan Greengrass CLI untuk mengembangkan dan men-debug komponen lokal.

> **Important**
>
> Kami menyarankan Anda menggunakan komponen ini hanya di lingkungan pengembangan, bukan lingkungan produksi. Komponen ini menyediakan akses ke informasi dan operasi yang biasanya tidak Anda perlukan di lingkungan produksi. Ikuti prinsip hak istimewa paling sedikit dengan menerapkan komponen ini hanya ke perangkat inti di mana Anda membutuhkannya.

Default: `false`

`INIT_CONFIG`

(Opsional) Path ke file konfigurasi yang akan digunakan untuk menginstal perangkat lunak inti AWS IoT Greengrass. Anda dapat menggunakan opsi ini untuk mengatur perangkat inti Greengrass baru dengan konfigurasi inti tertentu, atau untuk menentukan sumber daya yang ditetapkan secara manual, misalnya. Anda harus memasang file konfigurasi Anda ke jalur yang Anda tentukan dalam argumen ini.

`TRUSTED_PLUGIN`

Fitur ini tersedia untuk v2.4.0 dan yang lebih baru dari komponen inti Greengrass.

(Opsional) Jalur ke file JAR untuk dimuat sebagai plugin tepercaya. Gunakan opsi ini untuk menyediakan file JAR plugin penyediaan, seperti menginstal dengan penyediaan armada atau penyediaan khusus.

`THING_POLICY_NAME`

Fitur ini tersedia untuk v2.4.0 dan yang lebih baru dari komponen inti Greengrass.

(Opsional) Nama AWS IoT kebijakan untuk dilampirkan ke sertifikat AWS IoT benda perangkat inti ini. Jika AWS IoT kebijakan dengan nama ini tidak ada di perangkat lunak AWS IoT Greengrass Core Anda Akun AWS membuatnya.

Anda harus menentukan `PROVISION=true` untuk menerapkan argumen ini.

> **Note**
>
> Perangkat lunak AWS IoT Greengrass Core membuat AWS IoT kebijakan permisif secara default. Anda dapat menjelaskan kebijakan ini, atau membuat kebijakan khusus yang membatasi izin untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2.

Tentukan dependensi yang akan dipasang

Instruksi RUN di Dockerfile AWS IoT Greengrass mempersiapkan lingkungan kontainer untuk menjalankan penginstal perangkat lunak inti AWS IoT Greengrass. Anda dapat menyesuaikan dependensi yang diinstal sebelum penginstal perangkat lunak inti AWS IoT Greengrass berjalan di kontainer Docker.

Bangun citra AWS IoT Greengrass

Gunakan Dockerfile AWS IoT Greengrass untuk membangun citra kontainer AWS IoT Greengrass. Anda dapat menggunakan CLI Docker atau Docker Compose CLI untuk membangun citra dan memulai kontainer tersebut. Anda juga dapat menggunakan CLI Docker untuk membangun citra dan kemudian menggunakan Docker Compose untuk memulai kontainer Anda dari citra tersebut.

Docker

1. Pada mesin host, jalankan perintah berikut untuk beralih ke direktori yang berisi Dockerfile yang dikonfigurasi.

   ```
   cd download-directory/aws-greengrass-docker-nucleus-version
   ```

2. Jalankan perintah berikut untuk membangun gambar kontainer AWS IoT Greengrass dari Dockerfile.

   ```
   sudo docker build -t "platform/aws-iot-greengrass:nucleus-version" ./
   ```

Docker Compose

1. Pada mesin host, jalankan perintah berikut untuk beralih ke direktori yang berisi Dockerfile dan file Compose.

   ```
   cd download-directory/aws-greengrass-docker-nucleus-version
   ```

2. Jalankan perintah berikut untuk menggunakan file Compose untuk membangun citra kontainer AWS IoT Greengrass.

   ```
   docker-compose -f docker-compose.yml build
   ```

Anda telah berhasil membuat citra kontainer AWS IoT Greengrass. Gambar Docker memiliki perangkat lunak inti AWS IoT Greengrass yang terinstal. Anda sekarang dapat menjalankan perangkat lunak inti AWS IoT Greengrass dalam kontainer Docker

Jalankan AWS IoT Greengrass dalam kontainer Docker dengan penyediaan sumber daya otomatis

Tutorial ini menunjukkan cara menginstal dan menjalankan perangkat lunak AWS IoT Greengrass Core dalam wadah Docker dengan AWS sumber daya yang disediakan secara otomatis dan alat pengembangan lokal. Anda dapat menggunakan lingkungan pengembangan ini untuk mengeksplorasi fitur AWS IoT Greengrass dalam kontainer Docker. Perangkat lunak ini membutuhkan AWS kredensyal untuk menyediakan sumber daya ini dan menyebarkan alat pengembangan lokal.

Jika Anda tidak dapat memberikan AWS kredensyal ke penampung, Anda dapat menyediakan AWS sumber daya yang dibutuhkan perangkat inti untuk beroperasi. Anda juga dapat men-deploy alat pengembangan ke perangkat inti untuk digunakan sebagai perangkat pengembangan. Ini memungkinkan Anda memberikan lebih sedikit izin ke perangkat saat menjalankan penampung. Untuk informasi selengkapnya, lihat Jalankan AWS IoT Greengrass dalam kontainer Docker dengan penyediaan sumber daya manual.

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut ini:

- Sebuah Akun AWS. Jika Anda tidak memilikinya, lihat Mengatur sebuah Akun AWS.
- Pengguna IAM AWS dengan izin untuk menyediakan AWS IoT dan sumber daya IAM untuk perangkat inti Greengrass. Penginstal perangkat lunak inti AWS IoT Greengrass menggunakan kredensial AWS Anda untuk secara otomatis menyediakan sumber daya ini. Untuk informasi tentang kebijakan IAM minimal untuk secara otomatis menyediakan sumber daya, lihatKebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya.
- Citra Docker AWS IoT Greengrass. Anda dapat membuat gambar dari AWS IoT Greengrass Dockerfile.
- Komputer host tempat Anda menjalankan wadah Docker harus memenuhi persyaratan berikut:
  - Sistem operasi berbasis Linux dengan koneksi internet.
  - Mesin Docker versi 18.09 atau yang lebih baru.
  - (Opsional) Docker Compose versi 1.22 atau yang lebih baru. Docker Compose diperlukan hanya jika Anda ingin menggunakan Docker Compose CLI untuk menjalankan gambar Docker Anda.

Konfigurasikan kredensial AWS Anda

Pada langkah ini, Anda membuat file kredensial pada komputer host yang berisi kredensial keamanan AWS. Ketika Anda menjalankan citra Docker AWS IoT Greengrass, Anda harus memasang folder yang berisi file kredensial ini ke `/root/.aws/` di kontainer Docker. Penginstal AWS IoT Greengrass menggunakan kredensial ini untuk menyediakan sumber daya di Akun AWS Anda. Untuk informasi tentang kebijakan IAM minimal yang diperlukan installer untuk secara otomatis menyediakan sumber daya, lihat Kebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya.

1. Ambil salah satu dari yang berikut ini.

   - Kredensial jangka panjang untuk pengguna IAM. Untuk informasi tentang cara mengambil kredensial jangka panjang, lihat Mengelola access key untuk pengguna IAM di Panduan Pengguna IAM.
   - (Disarankan) Kredensial sementara untuk IAM role: Untuk informasi tentang cara mengambil kredensial sementara, lihat Menggunakan kredensial keamanan sementara dengan AWS CLI di Panduan Pengguna IAM.

2. Buat folder tempat Anda menempatkan file kredensial Anda.

   ```
   mkdir ./greengrass-v2-credentials
   ```

3. Gunakan editor teks untuk membuat file konfigurasi bernama `credentials` di folder `./greengrass-v2-credentials`.

   Misalnya, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file `credentials`.

   ```
   nano ./greengrass-v2-credentials/credentials
   ```

4. Tambahkan kredensial AWS Anda ke file `credentials` dalam format berikut:

   ```
   [default]
   aws_access_key_id = AKIAIOSFODNN7EXAMPLE
   aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   aws_session_token
    = AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
   ```

   Sertakan `aws_session_token` untuk kredensial sementara saja.

> **Important**
>
> Hapus file kredensial dari komputer host setelah Anda memulai kontainer AWS IoT Greengrass. Jika Anda tidak menghapus file kredensial, maka kredensial AWS akan tetap dipasang di dalam kontainer tersebut. Untuk informasi selengkapnya, lihat Jalankan perangkat lunak inti AWS IoT Greengrass dalam kontainer.

Buat sebuah file lingkungan.

Tutorial ini menggunakan file lingkungan untuk mengatur variabel lingkungan yang akan diteruskan ke penginstal perangkat lunak inti AWS IoT Greengrass di dalam kontainer Docker. Anda juga dapat menggunakan argumen `-e` atau `--env` di perintah `docker run` Anda untuk mengatur variabel lingkungan dalam kontainer Docker atau Anda dapat mengatur variabel dalam sebuah blok `environment` di file `docker-compose.yml`.

1. Gunakan editor teks untuk membuat file lingkungan bernama `.env`.

   Sebagai contoh, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat `.env` pada direktori saat ini.

   ```
   nano .env
   ```

2. Salin konten berikut ke dalam file tersebut.

   ```
   GGC_ROOT_PATH=/greengrass/v2
   AWS_REGION=region
   PROVISION=true
   THING_NAME=MyGreengrassCore
   THING_GROUP_NAME=MyGreengrassCoreGroup
   TES_ROLE_NAME=GreengrassV2TokenExchangeRole
   TES_ROLE_ALIAS_NAME=GreengrassCoreTokenExchangeRoleAlias
   COMPONENT_DEFAULT_USER=ggc_user:ggc_group
   ```

   Anda harus mengganti nilai berikut.

   - */greengrass/v2*. Folder root Greengrass yang ingin Anda gunakan untuk instalasi. Anda dapat menggunakan variabel lingkungan `GGC_ROOT` untuk menetapkan nilai ini.
   - *region*. Wilayah AWS tempat Anda membuat sumber daya.

- *MyGreengrassCore*. Nama objek AWS IoT Jika objek itu tidak ada, installer akan membuatnya. Penginstal mengunduh sertifikat tersebut untuk diautentikasi sebagai objek AWS IoT.
- *MyGreengrassCoreGroup*. Nama grup objek AWS IoT Jika grup objek tidak ada, installer akan membuatnya dan menambahkan objek itu padanya. Jika grup objek ada dan memiliki deployment yang aktif, perangkat inti akan men-download dan menjalankan perangkat lunak yang ditetapkan oleh deployment.
- *GreenGrassV2 TokenExchangeRole*. Ganti dengan nama peran pertukaran token IAM yang memungkinkan perangkat inti Greengrass untuk mendapatkan kredensial AWS sementara. Jika peran tidak ada, penginstal membuatnya dan membuat serta melampirkan kebijakan bernama *TokenExchangeRoleGreenGrassV2* Access. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.
- *GreengrassCoreTokenExchangeRoleAlias*. Alias peran pertukaran token. Jika alias peran tidak ada, penginstal akan membuatnya dan mengarahkannya ke peran pertukaran token IAM yang Anda tentukan. Untuk informasi selengkapnya, silakan lihat

> **Note**
>
> Anda dapat mengatur variabel `DEPLOY_DEV_TOOLS` lingkungan `true` untuk menyebarkan komponen CLI Greengrass, yang memungkinkan Anda mengembangkan komponen khusus di dalam wadah Docker. Kami menyarankan Anda menggunakan komponen ini hanya di lingkungan pengembangan, bukan lingkungan produksi. Komponen ini menyediakan akses ke informasi dan operasi yang biasanya tidak Anda perlukan di lingkungan produksi. Ikuti prinsip hak istimewa paling sedikit dengan menerapkan komponen ini hanya ke perangkat inti di mana Anda membutuhkannya.

Jalankan perangkat lunak inti AWS IoT Greengrass dalam kontainer

Tutorial ini menunjukkan cara memulai gambar Docker yang Anda buat di wadah Docker. Anda dapat menggunakan Docker CLI atau Docker Compose CLI untuk menjalankan gambar perangkat lunak inti AWS IoT Greengrass dalam kontainer Docker.

Docker

1. Jalankan perintah berikut untuk memulai kontainer Docker.

```
docker run --rm --init -it --name docker-image \
 -v path/to/greengrass-v2-credentials:/root/.aws/:ro \
 --env-file .env \
 -p 8883 \
 your-container-image:version
```

Contoh perintah ini menggunakan argumen berikut untuk docker run:

- `--rm`. Membersihkan kontainer saat keluar.
- `--init`. Menggunakan proses init dalam kontainer.

  > **Note**
  >
  > Argumen `--init` diperlukan untuk mematikan perangkat lunak inti AWS IoT Greengrass saat Anda menghentikan kontainer Docker.

- `-it`. (Opsional) Menjalankan kontainer Docker di latar depan sebagai proses interaktif. Anda dapat mengganti ini dengan `-d` untuk menjalankan kontainer Docker dalam mode terpisah sebagai gantinya. Untuk informasi lebih lanjut, lihat: Terpisah vs latar depandalam dokumentasi Docker.
- `--name`. Menjalankan kontainer bernama `aws-iot-greengrass`
- `-v`. Memasang volume ke kontainer Docker untuk membuat file konfigurasi dan file sertifikat yang tersedia untuk AWS IoT Greengrass yang berjalan di dalam kontainer tersebut.
- `--env-file`. (Opsional) Menentukan file lingkungan untuk mengatur variabel lingkungan yang akan diteruskan ke penginstal perangkat lunak AWS IoT Greengrass Core di dalam wadah Docker. Argumen ini diperlukan hanya jika Anda membuat file lingkungan untuk mengatur variabel lingkungan. Jika Anda tidak membuat file lingkungan, Anda dapat menggunakan `--env` argumen untuk mengatur variabel lingkungan secara langsung di perintah run Docker Anda.
- `-p`. (Opsional) Menerbitkan port kontainer 8883 ke mesin host. Argumen ini diperlukan jika Anda ingin terhubung dan berkomunikasi melalui MQTT karena AWS IoT Greengrass menggunakan port 8883 untuk lalu lintas MQTT. Untuk membuka port lain, gunakan `-p` argumen tambahan.

> **Note**
>
> Untuk menjalankan kontainer Docker Anda dengan peningkatan keamanan, Anda dapat menggunakan `--cap-drop` dan `--cap-add` untuk secara selektif mengaktifkan kemampuan Linux untuk kontainer Anda. Untuk informasi lebih lanjut, lihat:Keistimewaan waktu aktif dan kemampuan Linux dalam dokumentasi Docker.

2. Hapus kredensyal dari `./greengrass-v2-credentials` perangkat host.

```
rm -rf ./greengrass-v2-credentials
```

> **Important**
>
> Anda menghapus kredensyal ini, karena mereka memberikan izin luas yang hanya dibutuhkan perangkat inti selama penyiapan. Jika Anda tidak menghapus kredensyal ini, komponen Greengrass dan proses lain yang berjalan di wadah dapat mengaksesnya. Jika Anda perlu memberikan AWS kredensyal ke komponen Greengrass, gunakan layanan pertukaran token. Untuk informasi selengkapnya, lihat Berinteraksilah dengan layanan AWS.

Docker Compose

1. Gunakan editor teks untuk membuat file Docker Compose dengan nama `docker-compose.yml`.

   Sebagai contoh, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat `docker-compose.yml` pada direktori saat ini.

```
nano docker-compose.yml
```

> **Note**
>
> Anda juga dapat mengunduh dan menggunakan versi terbaru dari file Tulis AWS yang disediakan dari. GitHub

2. Tambahkan konten berikut ini ke file Compose. File Anda akan terlihat seperti contoh berikut. Ganti *docker-image* dengan nama image Docker Anda.

```
version: '3.7'

services:
  greengrass:
    init: true
    container_name: aws-iot-greengrass
    image: docker-image
    volumes:
      - ./greengrass-v2-credentials:/root/.aws/:ro
    env_file: .env
    ports:
      - "8883:8883"
```

Parameter berikut dalam contoh file Compose ini bersifat opsional:

- `ports`—Menerbitkan port kontainer 8883 ke mesin host. Parameter ini diperlukan jika Anda ingin terhubung dan berkomunikasi melalui MQTT karena AWS IoT Greengrass menggunakan port 8883 untuk lalu lintas MQTT.
- `env_file`—Menentukan file lingkungan untuk mengatur variabel lingkungan yang akan diteruskan ke penginstal perangkat lunak AWS IoT Greengrass Core di dalam wadah Docker. Parameter ini diperlukan hanya jika Anda membuat file lingkungan untuk mengatur variabel lingkungan. Jika Anda tidak membuat file lingkungan, Anda dapat menggunakan parameter lingkungan untuk menyetel variabel secara langsung di file Compose Anda.

> **Note**
>
> Untuk menjalankan kontainer Docker Anda dengan peningkatan keamanan, Anda dapat menggunakan `cap_drop` dan `cap_add` dalam file Compose Anda untuk secara selektif mengaktifkan kemampuan Linux untuk kontainer Anda. Untuk informasi lebih lanjut, lihat: Keistimewaan waktu aktif dan kemampuan Linux dalam dokumentasi Docker.

3. Jalankan perintah berikut untuk memulai kontainer Docker.

```
docker-compose -f docker-compose.yml up
```

4. Hapus kredensyal dari `./greengrass-v2-credentials` perangkat host.

```
rm -rf ./greengrass-v2-credentials
```

> **Important**
>
> Anda menghapus kredensyal ini, karena mereka memberikan izin luas yang hanya dibutuhkan perangkat inti selama penyiapan. Jika Anda tidak menghapus kredensyal ini, komponen Greengrass dan proses lain yang berjalan di wadah dapat mengaksesnya. Jika Anda perlu memberikan AWS kredensyal ke komponen Greengrass, gunakan layanan pertukaran token. Untuk informasi selengkapnya, lihat Berinteraksilah dengan layanan AWS.

Langkah selanjutnya

Perangkat lunak inti AWS IoT Greengrass sekarang berjalan dalam kontainer Docker Jalankan perintah berikut untuk mengambil ID kontainer untuk kontainer yang sedang berjalan.

```
docker ps
```

Anda kemudian dapat menjalankan perintah berikut untuk mengakses kontainer dan menjelajahi perangkat lunak inti AWS IoT Greengrass yang berjalan di dalam kontainer tersebut.

```
docker exec -it container-id /bin/bash
```

Untuk informasi tentang membuat komponen sederhana, lihat Langkah 4: Kembangkan dan uji komponen di perangkat Anda di Tutorial: Memulai dengan AWS IoT Greengrass V2

> **Note**
>
> Saat Anda menggunakan `docker exec` untuk menjalankan perintah di dalam kontainer Docker, perintah tersebut tidak tercatat di log Docker. Untuk mencatat perintah Anda di log Docker, lampirkan shell interaktif ke kontainer Docker. Untuk informasi selengkapnya, lihat Lampirkan shell interaktif ke kontainer Docker.

Berkas log inti AWS IoT Greengrass disebut `greengrass.log` dan terletak di */greengrass/v2/*`logs`. File log komponen juga terletak di direktori yang sama. Untuk menyalin log Greengrass ke direktori sementara pada host, jalankan perintah berikut:

```
docker cp container-id:/greengrass/v2/logs /tmp/logs
```

Jika Anda ingin mempertahankan log setelah keluar kontainer atau telah dihapus, kami sarankan Anda hanya melakukan bind-mount pada direktori */greengrass/v2*`/logs` ke direktori log sementara pada host dan bukan memasang seluruh direktori Greengrass. Untuk informasi selengkapnya, lihat Pertahankan log Greengrass di luar kontainer Docker.

Untuk berhenti menjalankan kontainer Docker AWS IoT Greengrass, jalankan `docker stop` atau `docker-compose -f docker-compose.yml stop`. Tindakan ini akan mengirimkan `SIGTERM` ke proses Greengrass dan menutup semua proses terkait yang dimulai dalam kontainer tersebut. Kontainer Docker diinisialisasi dengan executable `docker-init` sebagai proses PID 1, yang membantu dalam menghapus proses zombie sisa. Untuk informasi selengkapnya, lihat Menentukan proses init dalam dokumentasi Docker.

Untuk informasi tentang pemecahan masalah dengan menjalankan AWS IoT Greengrass dalam kontainer Docker, lihat Penyelesaian masalah AWS IoT Greengrass di kontainer Docker.

Jalankan AWS IoT Greengrass dalam kontainer Docker dengan penyediaan sumber daya manual

Tutorial ini menunjukkan cara menginstal dan menjalankan perangkat lunak inti AWS IoT Greengrass dalam kontainer Docker dengan sumber daya AWS yang disediakan secara manual.

Topik

- Prasyarat
- Ambil titik akhir AWS IoT
- Buat objek AWS IoT
- Buat sertifikat benda
- Konfigurasikan sertifikat benda
- Buat peran pertukaran token
- Unduh sertifikat ke perangkat
- Buat file konfigurasi

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut:

- Sebuah Akun AWS. Jika Anda tidak memilikinya, lihat Mengatur sebuah Akun AWS.
- Citra Docker AWS IoT Greengrass. Anda dapat membuat gambar dari AWS IoT Greengrass Dockerfile.
- Komputer host tempat Anda menjalankan wadah Docker harus memenuhi persyaratan berikut:
  - Sistem operasi berbasis Linux dengan koneksi internet.
  - Mesin Docker versi 18.09 atau yang lebih baru.
  - (Opsional) Docker Compose versi 1.22 atau yang lebih baru. Docker Compose diperlukan hanya jika Anda ingin menggunakan Docker Compose CLI untuk menjalankan gambar Docker Anda.

Ambil titik akhir AWS IoT

Dapatkan titik akhir AWS IoT untuk Akun AWS, dan simpan untuk digunakan nanti. Perangkat Anda menggunakan titik akhir ini untuk tersambung ke AWS IoT. Lakukan hal-hal berikut:

1. Dapatkan titik akhir data AWS IoT untuk perangkat Akun AWS Anda.

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

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Dapatkan titik akhir kredensial AWS IoT untuk Akun AWS Anda.

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

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "endpointAddress": "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
}
```

Buat objek AWS IoT

Objek AWS IoT mewakili perangkat dan entitas logis yang terhubung ke AWS IoT. Perangkat inti Greengrass merupakan objek AWS IoT. Saat Anda mendaftarkan suatu perangkat sebagai objek AWS IoT, perangkat itu dapat menggunakan sertifikat digital untuk diautentikasi dengan AWS.

Di bagian ini, Anda membuat AWS IoT sesuatu yang mewakili perangkat Anda.

Untuk membuat objek AWS IoT

1. Buat objek AWS IoT untuk perangkat Anda. Pada komputer pengembangan Anda, jalankan perintah berikut.

   - Ganti *MyGreengrassCore*dengan nama benda yang akan digunakan. Nama ini juga merupakan nama perangkat inti Greengrass Anda.

     > **Note**
     >
     > Nama objek tidak dapat berisi karakter titik dua (`:`).

   ```
   aws iot create-thing --thing-name MyGreengrassCore
   ```

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "thingName": "MyGreengrassCore",
     "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
     "thingId": "8cb4b6cd-268e-495d-b5b9-1713d71dbf42"
   }
   ```

2. (Opsional) Tambahkan AWS IoT pada grup objek baru atau yang sudah ada. Anda menggunakan grup objek untuk mengelola armada perangkat inti Greengrass. Saat menerapkan

komponen perangkat lunak ke perangkat, Anda dapat menargetkan perangkat individual atau grup perangkat. Anda dapat menambahkan suatu perangkat ke grup objek dengan deployment Greengrass aktif untuk men-deploy komponen perangkat lunak grup objek tersebut ke perangkat. Lakukan hal-hal berikut:

a. (Opsional) Buat grup objek AWS IoT.

   - Ganti *MyGreengrassCoreGroup*dengan nama grup benda yang akan dibuat.

   > **Note**
   >
   > Nama grup objek tidak dapat berisi karakter titik dua (`:`).

   ```
   aws iot create-thing-group --thing-group-name MyGreengrassCoreGroup
   ```

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "thingGroupName": "MyGreengrassCoreGroup",
     "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
   MyGreengrassCoreGroup",
     "thingGroupId": "4df721e1-ff9f-4f97-92dd-02db4e3f03aa"
   }
   ```

b. Tambahkan objek AWS IoT pada grup objek.

   - Ganti *MyGreengrassCore*dengan nama AWS IoT benda Anda.
   - Ganti *MyGreengrassCoreGroup*dengan nama grup benda.

   ```
   aws iot add-thing-to-thing-group --thing-name MyGreengrassCore --thing-group-
   name MyGreengrassCoreGroup
   ```

   Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

Buat sertifikat benda

Saat Anda mendaftarkan perangkat sebagai objek AWS IoT, perangkat itu dapat menggunakan sertifikat digital untuk diautentikasi dengan AWS. Sertifikat ini memungkinkan perangkat untuk berkomunikasi dengan AWS IoT dan AWS IoT Greengrass.

Di bagian ini, Anda membuat dan mengunduh sertifikat yang dapat digunakan perangkat Anda untuk terhubungAWS.

Untuk membuat sertifikat benda

1. Buat folder di mana Anda mengunduh sertifikat untuk objek AWS IoT.

   ```
   mkdir greengrass-v2-certs
   ```

2. Buat dan unduh sertifikat tersebut untuk objek AWS IoT.

   ```
   aws iot create-keys-and-certificate --set-as-active --certificate-pem-outfile
    greengrass-v2-certs/device.pem.crt --public-key-outfile greengrass-v2-certs/
   public.pem.key --private-key-outfile greengrass-v2-certs/private.pem.key
   ```

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
   aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
     "certificateId":
    "aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
     "certificatePem": "-----BEGIN CERTIFICATE-----
   MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
    0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
    WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
    EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
    jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
    MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
    WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
    HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
    BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
    k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
    ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
    AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
    KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
   ```

```
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
  "keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----\
MIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\
MMEXAMPLEuuN/dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/gHr99VEEXAMPLE5VF13\
59VK7cEXAMPLE67GK+y+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQWE
\\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa\
hJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\
FQIDAQAB\
-----END PUBLIC KEY-----\
",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\
key omitted for security reasons\
-----END RSA PRIVATE KEY-----\
"
  }
}
```

Simpan Nama Sumber Daya Amazon (ARN) sertifikat yang akan digunakan untuk mengonfigurasi sertifikat nanti.

Konfigurasikan sertifikat benda

Lampirkan sertifikat benda ke AWS IoT benda yang Anda buat sebelumnya, dan tambahkan AWS IoT kebijakan ke sertifikat untuk menentukan AWS IoT izin untuk perangkat inti.

Untuk mengkonfigurasi sertifikat benda

1. Lampirkan sertifikat pada objek AWS IoT.

   - Ganti *MyGreengrassCore*dengan nama AWS IoT benda Anda.
   - Ganti Amazon Resource Name (ARN) sertifikat dengan ARN sertifikat yang Anda buat pada langkah sebelumnya.

```
aws iot attach-thing-principal --thing-name MyGreengrassCore
 --principal arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

2. Buat dan lampirkan kebijakan AWS IoT yang menentukan izin AWS IoT untuk perangkat inti Greengrass Anda. Kebijakan berikut memungkinkan akses ke semua topik MQTT dan operasi Greengrass, sehingga perangkat Anda bekerja dengan aplikasi kustom dan perubahan di masa mendatang yang memerlukan operasi Greengrass baru. Anda dapat membatasi kebijakan ini berdasarkan kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2.

   Jika Anda telah menyiapkan perangkat inti Greengrass sebelumnya, Anda dapat melampirkan kebijakan AWS IoT alih-alih menciptakan kebijakan baru.

   Lakukan hal-hal berikut:

   a. Buat file yang berisi dokumen AWS IoT kebijakan yang dibutuhkan perangkat inti Greengrass.

      Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

      ```
      nano greengrass-v2-iot-policy.json
      ```

      Salin JSON berikut ke dalam file.

      ```
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": [
              "iot:Publish",
              "iot:Subscribe",
              "iot:Receive",
              "iot:Connect",
              "greengrass:*"
            ],
            "Resource": [
              "*"
            ]
          }
        ]
      ```

```
}
```

b. Buat AWS IoT kebijakan dari dokumen kebijakan.

- Ganti *GreenGrassV2IoT ThingPolicy* dengan nama kebijakan yang akan dibuat.

```
aws iot create-policy --policy-name GreengrassV2IoTThingPolicy --policy-
document file://greengrass-v2-iot-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "policyName": "GreengrassV2IoTThingPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
  "policyDocument": "{
    \\"Version\\": \\"2012-10-17\\",
    \\"Statement\\": [
      {
        \\"Effect\\": \\"Allow\\",
        \\"Action\\": [
          \\"iot:Publish\\",
          \\"iot:Subscribe\\",
          \\"iot:Receive\\",
          \\"iot:Connect\\",
          \\"greengrass:*\\"
        ],
        \\"Resource\\": [
          \\"*\\"
        ]
      }
    ]
  }",
  "policyVersionId": "1"
}
```

c. Lampirkan kebijakan AWS IoT ke sertifikat objek AWS IoT.

- Ganti *GreenGrassV2IoT ThingPolicy* dengan nama kebijakan yang akan dilampirkan.
- Ganti ARN target dengan ARN sertifikat untuk objek AWS IoT Anda.

```
aws iot attach-policy --policy-name GreengrassV2IoTThingPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

Buat peran pertukaran token

Perangkat inti Greengrass menggunakan peran layanan IAM, yang disebut peran pertukaran token, untuk mengotorisasi panggilan ke layanan AWS. Perangkat menggunakan penyedia AWS IoT kredensional untuk mendapatkan AWS kredensi sementara untuk peran ini, yang memungkinkan perangkat berinteraksi, mengirim log ke Amazon LogAWS IoT, dan mengunduh CloudWatch artefak komponen khusus dari Amazon S3. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Anda menggunakan alias AWS IoT peran untuk mengonfigurasi peran pertukaran token untuk perangkat inti Greengrass. Alias peran memungkinkan Anda mengubah peran pertukaran token untuk suatu perangkat tetapi menjaga konfigurasi perangkat tetap sama. Untuk informasi selengkapnya, lihat Mengotorisasi panggilan langsung ke layanan AWS di Panduan Developer AWS IoT Core.

Pada bagian ini, Anda membuat pertukaran token IAM role dan alias peran AWS IoT yang menunjuk ke peran tersebut. Jika Anda telah menyiapkan perangkat inti Greengrass, Anda dapat menggunakan peran pertukaran token dan alias peran alih-alih membuat yang baru. Kemudian, Anda mengonfigurasi objek AWS IoT untuk menggunakan peran dan alias itu.

Buat peran pertukaran token IAM role

1. Buat peran IAM yang dapat digunakan perangkat Anda sebagai peran pertukaran token. Lakukan hal-hal berikut:

   a. Buat file yang berisi dokumen kebijakan kepercayaan yang memerlukan peran pertukaran token.

      Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano device-role-trust-policy.json
```

Salin JSON berikut ke dalam file.

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

b. Buat peran pertukaran token dengan dokumen kebijakan kepercayaan.

- Ganti *GreenGrassV2 TokenExchangeRole* dengan nama peran IAM yang akan dibuat.

```
aws iam create-role --role-name GreengrassV2TokenExchangeRole --assume-role-
policy-document file://device-role-trust-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "Role": {
    "Path": "/",
    "RoleName": "GreengrassV2TokenExchangeRole",
    "RoleId": "AROAZ2YMUHYHK5OKM77FB",
    "Arn": "arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole",
    "CreateDate": "2021-02-06T00:13:29+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
```

```
            "Service": "credentials.iot.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
```

c. Buat file yang berisi dokumen kebijakan akses yang diperlukan oleh peran pertukaran token.

Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano device-role-access-policy.json
```

Salin JSON berikut ke dalam file.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Kebijakan akses ini tidak mengizinkan akses ke artefak komponen dalam bucket S3. Untuk men-deploy komponen kustom yang menentukan artefak di Amazon S3, Anda harus menambahkan izin untuk peran tersebut untuk memungkinkan perangkat inti Anda untuk mengambil artefak komponen. Untuk informasi selengkapnya, lihat Izinkan akses ke bucket S3 untuk artefak komponen.

Jika Anda belum memiliki bucket S3 untuk artefak komponen, Anda dapat menambahkan izin ini nanti setelah membuat bucket.

d. Buat kebijakan IAM dari dokumen kebijakan.

- Ganti *GreenGrassV2 TokenExchangeRoleAccess* dengan nama kebijakan IAM yang akan dibuat.

```
aws iam create-policy --policy-name GreengrassV2TokenExchangeRoleAccess --
policy-document file://device-role-access-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "Policy": {
    "PolicyName": "GreengrassV2TokenExchangeRoleAccess",
    "PolicyId": "ANPAZ2YMUHYHACI7C5Z66",
    "Arn": "arn:aws:iam::123456789012:policy/
GreengrassV2TokenExchangeRoleAccess",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2021-02-06T00:37:17+00:00",
    "UpdateDate": "2021-02-06T00:37:17+00:00"
  }
}
```

e. Lampirkan kebijakan IAM untuk peran pertukaran token.

- Ganti *GreenGrassV2 TokenExchangeRole* dengan nama peran IAM.
- Ganti ARN peran dengan ARN dari kebijakan IAM yang Anda buat di langkah sebelumnya.

```
aws iam attach-role-policy --role-name GreengrassV2TokenExchangeRole --policy-
arn arn:aws:iam::123456789012:policy/GreengrassV2TokenExchangeRoleAccess
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

2. Buat alias AWS IoT peran yang menunjuk ke peran pertukaran token.

   - Ganti *GreengrassCoreTokenExchangeRoleAlias*dengan nama alias peran yang akan dibuat.
   - Ganti ARN peran dengan ARN dari IAM role yang Anda buat di langkah sebelumnya.

   ```
   aws iot create-role-alias --role-alias GreengrassCoreTokenExchangeRoleAlias --role-
   arn arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole
   ```

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "roleAlias": "GreengrassCoreTokenExchangeRoleAlias",
     "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
   GreengrassCoreTokenExchangeRoleAlias"
   }
   ```

   > **Note**
   >
   > Untuk membuat alias peran, Anda harus memiliki izin untuk melewati IAM role pertukaran token ke AWS IoT. Jika Anda menerima pesan galat saat mencoba membuat alias peran, periksa apakah AWS pengguna Anda memiliki izin ini. Untuk informasi lebih lanjut, lihat Memberikan izin pengguna untuk meneruskan peran ke layanan AWS di Panduan Pengguna AWS Identity and Access Management.

3. Buat dan lampirkan kebijakan AWS IoT yang memungkinkan perangkat inti Greengrass Anda untuk menggunakan alias peran untuk meneruskan peran pertukaran token. Jika Anda telah menyiapkan perangkat inti Greengrass sebelumnya, Anda dapat melampirkan kebijakan AWS IoT pada alias perannya alih-alih menciptakan kebijakan baru. Lakukan hal-hal berikut:

   a. (Opsional) Buat file yang berisi dokumen kebijakan AWS IoT yang diperlukan oleh alias peran tersebut.

      Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

      ```
      nano greengrass-v2-iot-role-alias-policy.json
      ```

Salin JSON berikut ke dalam file.

- Ganti ARN sumber daya dengan ARN alias peran Anda.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:AssumeRoleWithCertificate",
      "Resource": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
    }
  ]
}
```

b. Buat AWS IoT kebijakan dari dokumen kebijakan.

- Ganti *GreengrassCoreTokenExchangeRoleAliasPolicy*dengan nama AWS IoT kebijakan yang akan dibuat.

```
aws iot create-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --policy-document file://greengrass-v2-iot-role-alias-policy.json
```

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "policyName": "GreengrassCoreTokenExchangeRoleAliasPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassCoreTokenExchangeRoleAliasPolicy",
  "policyDocument": "{
    \\"Version\\":\\"2012-10-17\\",
    \\"Statement\\": [
      {
        \\"Effect\\": \\"Allow\\",
        \\"Action\\": \\"iot:AssumeRoleWithCertificate\\",
        \\"Resource\\": \\"arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias\\"
      }
    ]
```

```
  }",
  "policyVersionId": "1"
}
```

c. Lampirkan kebijakan AWS IoT ke sertifikat objek AWS IoT.

- Ganti *GreengrassCoreTokenExchangeRoleAliasPolicy*dengan nama AWS IoT kebijakan alias peran.
- Ganti ARN target dengan ARN sertifikat untuk objek AWS IoT Anda.

```
aws iot attach-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

Unduh sertifikat ke perangkat

Sebelumnya, Anda mengunduh sertifikat perangkat ke komputer pengembangan Anda. Di bagian ini, Anda mengunduh sertifikat otoritas sertifikat root Amazon (CA). Kemudian, jika Anda berencana untuk menjalankan perangkat lunak AWS IoT Greengrass Core di Docker pada komputer yang berbeda dari komputer pengembangan Anda, Anda menyalin sertifikat ke komputer host itu. Perangkat lunak AWS IoT Greengrass Core menggunakan sertifikat ini untuk terhubung ke layanan AWS IoT cloud.

Untuk mengunduh sertifikat ke perangkat

1. Di komputer pengembangan Anda, unduh sertifikat otoritas sertifikat root Amazon (CA). AWS IoTsertifikat dikaitkan dengan sertifikat CA root Amazon secara default.

   Linux or Unix

   ```
   sudo curl -o ./greengrass-v2-certs/AmazonRootCA1.pem https://
   www.amazontrust.com/repository/AmazonRootCA1.pem
   ```

Windows Command Prompt (CMD)

```
curl -o .\greengrass-v2-certs\AmazonRootCA1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem
```

PowerShell

```
iwr -Uri https://www.amazontrust.com/repository/AmazonRootCA1.pem -OutFile .
\greengrass-v2-certs\AmazonRootCA1.pem
```

2. Jika Anda berencana untuk menjalankan perangkat lunak AWS IoT Greengrass Core di Docker pada perangkat yang berbeda dari komputer pengembangan Anda, salin sertifikat ke komputer host. Jika SSH dan SCP diaktifkan pada komputer pengembangan dan komputer host, Anda dapat menggunakan `scp` perintah di komputer pengembangan Anda untuk mentransfer sertifikat. Ganti *`device-ip-address`*dengan alamat IP komputer host Anda.

   ```
   scp -r greengrass-v2-certs/ device-ip-address:~
   ```

Buat file konfigurasi

1. Di komputer host, buat folder tempat Anda menempatkan file konfigurasi Anda.

   ```
   mkdir ./greengrass-v2-config
   ```

2. Gunakan editor teks untuk membuat file konfigurasi bernama `config.yaml` di folder `./greengrass-v2-config`.

   Misalnya, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat `config.yaml`.

   ```
   nano ./greengrass-v2-config/config.yaml
   ```

3. Salin konten YAML berikut ke dalam file tersebut. File konfigurasi parsial ini menentukan parameter sistem dan parameter inti Greengrass.

   ```
   ---
   system:
     certificateFilePath: "/tmp/certs/device.pem.crt"
   ```

```
  privateKeyPath: "/tmp/certs/private.pem.key"
  rootCaPath: "/tmp/certs/AmazonRootCA1.pem"
  rootpath: "/greengrass/v2"
  thingName: "MyGreengrassCore"
services:
  aws.greengrass.Nucleus:
    componentType: "NUCLEUS"
    version: "nucleus-version"
    configuration:
      awsRegion: "region"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      iotDataEndpoint: "device-data-prefix-ats.iot.region.amazonaws.com"
      iotCredEndpoint: "device-credentials-prefix.credentials.region.amazonaws.com"
```

Kemudian, ganti nilai berikut:

- */tmp/certs*. Direktori dalam kontainer Docker yang padanya Anda pasang sertifikat yang diunduh ketika Anda memulai kontainer.
- */greengrass/v2*. Folder root Greengrass yang ingin Anda gunakan untuk instalasi. Anda dapat menggunakan variabel lingkungan `GGC_ROOT` untuk menetapkan nilai ini.
- *MyGreengrassCore*. Nama objek AWS IoT
- *nucleus-version*. Versi perangkat lunak inti AWS IoT Greengrass yang akan diinstal. Nilai ini harus sesuai dengan versi gambar Docker atau Dockerfile yang telah Anda download. Jika Anda telah mengunduh citra Docker Greengrass dengan tanda `latest`, gunakan **`docker inspect image-id`** untuk melihat versi citra tersebut.
- *region*. Wilayah AWS tempat Anda membuat sumber daya AWS IoT Anda. Anda juga harus menentukan nilai yang sama untuk variabel lingkungan `AWS_REGION` di file lingkungan.
- *GreengrassCoreTokenExchangeRoleAlias*. Alias peran pertukaran token.
- *device-data-prefix*. Awalan untuk titik akhir data AWS IoT Anda.
- *device-credentials-prefix*. Awalan untuk Titik akhir kredensial AWS IoT Anda.

Buat sebuah file lingkungan

Tutorial ini menggunakan file lingkungan untuk mengatur variabel lingkungan yang akan diteruskan ke penginstal perangkat lunak inti AWS IoT Greengrass di dalam kontainer Docker. Anda juga dapat menggunakan argumen `-e` atau `--env` di perintah `docker run` Anda untuk mengatur

variabel lingkungan dalam kontainer Docker atau Anda dapat mengatur variabel dalam sebuah blok `environment` di file `docker-compose.yml`.

1. Gunakan editor teks untuk membuat file lingkungan bernama `.env`.

   Sebagai contoh, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat `.env` pada direktori saat ini.

   ```
   nano .env
   ```

2. Salin konten berikut ke dalam file tersebut.

   ```
   GGC_ROOT_PATH=/greengrass/v2
   AWS_REGION=region
   PROVISION=false
   COMPONENT_DEFAULT_USER=ggc_user:ggc_group
   INIT_CONFIG=/tmp/config/config.yaml
   ```

   Anda harus mengganti nilai berikut.

   - */greengrass/v2*. Jalur ke folder root yang akan digunakan untuk menginstal perangkat lunak AWS IoT Greengrass Core.
   - *region*. Wilayah AWS tempat Anda membuat sumber daya AWS IoT Anda. Anda harus menentukan nilai yang sama untuk parameter konfigurasi `awsRegion` di file konfigurasi Anda.
   - */tmp/config/*. Folder tempat Anda memasang file konfigurasi saat Anda memulai wadah Docker.

   > **Note**
   >
   > Anda dapat mengatur variabel `DEPLOY_DEV_TOOLS` lingkungan `true` untuk menyebarkan komponen CLI Greengrass, yang memungkinkan Anda mengembangkan komponen khusus di dalam wadah Docker. Kami menyarankan Anda menggunakan komponen ini hanya di lingkungan pengembangan, bukan lingkungan produksi. Komponen ini menyediakan akses ke informasi dan operasi yang biasanya tidak Anda perlukan di lingkungan produksi. Ikuti prinsip hak istimewa paling sedikit dengan menerapkan komponen ini hanya ke perangkat inti di mana Anda membutuhkannya.

Jalankan perangkat lunak inti AWS IoT Greengrass dalam kontainer

Tutorial ini menunjukkan kepada Anda bagaimana memulai image Docker yang Anda bangun dalam wadah Docker. Anda dapat menggunakan Docker CLI atau Docker Compose CLI untuk menjalankan gambar perangkat lunak inti AWS IoT Greengrass dalam kontainer Docker.

Docker

- Tutorial ini menunjukkan cara memulai gambar Docker yang Anda buat di wadah Docker.

  ```
  docker run --rm --init -it --name docker-image \
   -v path/to/greengrass-v2-config:/tmp/config/:ro \
   -v path/to/greengrass-v2-certs:/tmp/certs:ro \
   --env-file .env \
   -p 8883 \
   your-container-image:version
  ```

  Contoh perintah ini menggunakan argumen berikut untuk docker run:

  - `--rm`. Membersihkan kontainer saat keluar.
  - `--init`. Menggunakan proses init dalam kontainer.

    > **Note**
    >
    > Argumen `--init` diperlukan untuk mematikan perangkat lunak inti AWS IoT Greengrass saat Anda menghentikan kontainer Docker.

  - `-it`. (Opsional) Menjalankan kontainer Docker di latar depan sebagai proses interaktif. Anda dapat mengganti ini dengan `-d` untuk menjalankan kontainer Docker dalam mode terpisah sebagai gantinya. Untuk informasi lebih lanjut, lihat: Terpisah vs latar depandalam dokumentasi Docker.
  - `--name`. Menjalankan kontainer bernama `aws-iot-greengrass`
  - `-v`. Memasang volume ke kontainer Docker untuk membuat file konfigurasi dan file sertifikat yang tersedia untuk AWS IoT Greengrass yang berjalan di dalam kontainer tersebut.
  - `--env-file`. (Opsional) Menentukan file lingkungan untuk mengatur variabel lingkungan yang akan diteruskan ke penginstal perangkat lunak AWS IoT Greengrass Core di dalam wadah Docker. Argumen ini diperlukan hanya jika Anda membuat file lingkungan untuk

mengatur variabel lingkungan. Jika Anda tidak membuat file lingkungan, Anda dapat menggunakan `--env` argumen untuk mengatur variabel lingkungan secara langsung di perintah run Docker Anda.

- `-p`. (Opsional) Menerbitkan port kontainer 8883 ke mesin host. Argumen ini diperlukan jika Anda ingin terhubung dan berkomunikasi melalui MQTT karena AWS IoT Greengrass menggunakan port 8883 untuk lalu lintas MQTT. Untuk membuka port lain, gunakan `-p` argumen tambahan.

> **Note**
>
> Untuk menjalankan kontainer Docker Anda dengan peningkatan keamanan, Anda dapat menggunakan `--cap-drop` dan `--cap-add` untuk secara selektif mengaktifkan kemampuan Linux untuk kontainer Anda. Untuk informasi lebih lanjut, lihat: Keistimewaan waktu aktif dan kemampuan Linux dalam dokumentasi Docker.

Docker Compose

1. Gunakan editor teks untuk membuat file Docker Compose dengan nama `docker-compose.yml`.

   Sebagai contoh, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat `docker-compose.yml` pada direktori saat ini.

   ```
   nano docker-compose.yml
   ```

   > **Note**
   >
   > Anda juga dapat mengunduh dan menggunakan versi terbaru dari file Tulis AWS yang disediakan dari. GitHub

2. Tambahkan konten berikut ini ke file Compose. File Anda akan terlihat seperti contoh berikut. Ganti: versi *`your-container-name`* *`dengan nama gambar`* Docker Anda.

   ```
   version: '3.7'

   services:
   ```

```
  greengrass:
    init: true
    build:
      context: .
    container_name: aws-iot-greengrass
    image: your-container-name:version
    volumes:
      - /path/to/greengrass-v2-config:/tmp/config/:ro
      - /path/to/greengrass-v2-certs:/tmp/certs:ro
    env_file: .env
    ports:
      - "8883:8883"
```

Parameter berikut dalam contoh file Compose ini bersifat opsional:

- `ports`—Menerbitkan port kontainer 8883 ke mesin host. Parameter ini diperlukan jika Anda ingin terhubung dan berkomunikasi melalui MQTT karena AWS IoT Greengrass menggunakan port 8883 untuk lalu lintas MQTT.
- `env_file`—Menentukan file lingkungan untuk mengatur variabel lingkungan yang akan diteruskan ke penginstal perangkat lunak AWS IoT Greengrass Core di dalam wadah Docker. Parameter ini diperlukan hanya jika Anda membuat file lingkungan untuk mengatur variabel lingkungan. Jika Anda tidak membuat file lingkungan, Anda dapat menggunakan parameter lingkungan untuk menyetel variabel secara langsung di file Compose Anda.

> **Note**
>
> Untuk menjalankan kontainer Docker Anda dengan peningkatan keamanan, Anda dapat menggunakan `cap_drop` dan `cap_add` dalam file Compose Anda untuk secara selektif mengaktifkan kemampuan Linux untuk kontainer Anda. Untuk informasi lebih lanjut, lihat: Keistimewaan waktu aktif dan kemampuan Linux dalam dokumentasi Docker.

3. Jalankan perintah berikut untuk memulai kontainer.

```
docker-compose -f docker-compose.yml up
```

Langkah selanjutnya

Perangkat lunak inti AWS IoT Greengrass sekarang berjalan dalam kontainer Docker Jalankan perintah berikut untuk mengambil ID kontainer untuk kontainer yang sedang berjalan.

```
docker ps
```

Anda kemudian dapat menjalankan perintah berikut untuk mengakses kontainer dan menjelajahi perangkat lunak inti AWS IoT Greengrass yang berjalan di dalam kontainer tersebut.

```
docker exec -it container-id /bin/bash
```

Untuk informasi tentang membuat komponen sederhana, lihat Langkah 4: Kembangkan dan uji komponen di perangkat Anda di Tutorial: Memulai dengan AWS IoT Greengrass V2

> **Note**
>
> Saat Anda menggunakan `docker exec` untuk menjalankan perintah di dalam kontainer Docker, perintah tersebut tidak tercatat di log Docker. Untuk mencatat perintah Anda di log Docker, lampirkan shell interaktif ke kontainer Docker. Untuk informasi selengkapnya, lihat Lampirkan shell interaktif ke kontainer Docker.

Berkas log inti AWS IoT Greengrass disebut `greengrass.log` dan terletak di */greengrass/v2/*`logs`. File log komponen juga terletak di direktori yang sama. Untuk menyalin log Greengrass ke direktori sementara pada host, jalankan perintah berikut:

```
docker cp container-id:/greengrass/v2/logs /tmp/logs
```

Jika Anda ingin mempertahankan log setelah keluar kontainer atau telah dihapus, kami sarankan Anda hanya melakukan bind-mount pada direktori */greengrass/v2*`/logs` ke direktori log sementara pada host dan bukan memasang seluruh direktori Greengrass. Untuk informasi selengkapnya, lihat Pertahankan log Greengrass di luar kontainer Docker.

Untuk berhenti menjalankan kontainer Docker AWS IoT Greengrass, jalankan `docker stop` atau `docker-compose -f docker-compose.yml stop`. Tindakan ini akan mengirimkan `SIGTERM` ke proses Greengrass dan menutup semua proses terkait yang dimulai dalam kontainer tersebut. Kontainer Docker diinisialisasi dengan executable `docker-init` sebagai proses PID 1, yang

membantu dalam menghapus proses zombie sisa. Untuk informasi selengkapnya, lihat Menentukan proses init dalam dokumentasi Docker.

Untuk informasi tentang pemecahan masalah dengan menjalankan AWS IoT Greengrass dalam kontainer Docker, lihat Penyelesaian masalah AWS IoT Greengrass di kontainer Docker.

Penyelesaian masalah AWS IoT Greengrass di kontainer Docker

Gunakan informasi berikut untuk membantu Anda memecahkan masalah dengan menjalankan AWS IoT Greengrass dalam kontainer Docker dan untuk men-debug masalah dengan AWS IoT Greengrass dalam kontainer Docker.

Topik

- Memecahkan masalah dengan menjalankan kontainer Docker
- Debugging AWS IoT Greengrass di kontainer Docker

Memecahkan masalah dengan menjalankan kontainer Docker

Gunakan informasi berikut untuk membantu menyelesaikan masalah dengan menjalankan AWS IoT Greengrass di kontainer Docker.

Topik

- Kesalahan: Tidak dapat melakukan login interaktif dari perangkat non TTY
- Kesalahan: Opsi tidak dikenal: - no-include-email
- Kesalahan: Firewall memblokir berbagi file antara windows dan kontainer.
- Kesalahan: Terjadi kesalahan (AccessDeniedException) saat memanggil GetAuthorizationToken operasi: Pengguna: arn:aws:iam:: account-id:user/ <user-name>tidak diizinkan untuk melakukan: ecr: on resource: * GetAuthorizationToken
- Kesalahan: Anda telah mencapai batas kecepatan tarik

Kesalahan: Tidak dapat melakukan login interaktif dari perangkat non TTY

Kesalahan ini dapat terjadi ketika Anda menjalankan perintah `aws ecr get-login-password`. Pastikan Anda telah menginstal AWS CLI versi 2 atau versi 1. 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 dikenal: - no-include-email

Kesalahan ini dapat terjadi ketika Anda menjalankan perintah `aws ecr get-login`. Pastikan Anda memiliki versi AWS CLI terkini yang diinstal (misalnya, jalankan: `pip install awscli --upgrade --user`). Untuk informasi lebih lanjut, lihat: Menginstal AWS Command Line Interface di Microsoft Windows di Panduan Pengguna AWS Command Line Interface.

Kesalahan: Firewall memblokir berbagi file antara windows dan kontainer.

Anda mungkin menerima kesalahan ini atau `Firewall Detected` saat menjalankan Docker di komputer Windows. Hal ini juga dapat terjadi jika Anda masuk pada jaringan pribadi virtual (VPN) dan pengaturan jaringan Anda mencegah dipasangnya drive bersama. Dalam situasi itu, matikan VPN dan jalankan kembali kontainer Docker.

Kesalahan: Terjadi kesalahan (AccessDeniedException) saat memanggil GetAuthorizationToken operasi: Pengguna: arn:aws:iam:: ***account-id:user/*** <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 lebih lanjut, lihat: Contoh Kebijakan Repositori Amazon ECR dan Mengakses Satu Repositori Amazon ECR di Panduan Pengguna Amazon ECR.

Kesalahan: Anda telah mencapai batas kecepatan tarik

Docker Hub membatasi jumlah permintaan tarik yang dapat dibuat oleh pengguna anonim dan Free Docker Hub. Jika Anda melebihi batas kecepatan tersebut untuk permintaan tarik pengguna anonim atau gratis, Anda akan menerima salah satu dari kesalahan berikut:

```
ERROR: toomanyrequests: Too Many Requests.
```

```
You have reached your pull rate limit.
```

Untuk mengatasi kesalahan ini, Anda dapat menunggu selama beberapa jam sebelum Anda mencoba permintaan tarik lain. Jika Anda berencana untuk secara konsisten mengirimkan sejumlah besar permintaan tarik, lihat situs web Docker Hub untuk informasi tentang batas kecepatan, dan opsi untuk mengautentikasi dan meningkatkan akun Docker Anda.

Debugging AWS IoT Greengrass di kontainer Docker

Untuk men-debug masalah dengan kontainer Docker, Anda dapat mempertahankan log waktu aktif Greengrass atau melampirkan shell interaktif pada kontainer Docker.

Pertahankan log Greengrass di luar kontainer Docker

Setelah Anda menghentikan AWS IoT Greengrass, Anda dapat menggunakan perintah `docker cp` berikut untuk menyalin log Greengrass dari kontainer Docker ke direktori log sementara.

```
docker cp container-id:/greengrass/v2/logs /tmp/logs
```

Untuk mempertahankan log bahkan setelah kontainer keluar atau dihapus, Anda harus menjalankan kontainer Docker AWS IoT Greengrass setelah mengikat-pasang direktori */greengrass/v2*`/logs`.

Untuk melakukan ikat-pasang pada direktori */greengrass/v2*`/logs`, lakukan salah satu hal berikut ini ketika Anda menjalankan kontainer Docker AWS IoT Greengrass.

- Sertakan `-v` */tmp/logs*`:`*/greengrass/v2*`/logs:ro` di perintah `docker run`.

  Ubah blok `volumes` dalam file Compose untuk menyertakan baris berikut sebelum Anda menjalankan perintah `docker-compose up`.

  ```
  volumes:
   - /tmp/logs:/greengrass/v2/logs:ro
  ```

Anda kemudian dapat memeriksa log Anda di */tmp/logs* pada host Anda untuk melihat log Greengrass sementara AWS IoT Greengrass berjalan di dalam kontainer Docker.

Untuk informasi tentang menjalankan kontainer Docker Greengrass, lihat Jalankan AWS IoT Greengrass di Docker dengan penyediaan manual dan Jalankan AWS IoT Greengrass di Docker dengan penyediaan otomatis

Lampirkan shell interaktif ke kontainer Docker

Saat Anda menggunakan `docker exec` untuk menjalankan perintah di dalam kontainer Docker, perintah tersebut tidak tertangkap di log Docker. Pencatatan perintah Anda dalam log Docker dapat membantu Anda menyelidiki keadaan kontainer Docker Greengrass. Lakukan salah satu dari cara berikut:

- Jalankan perintah berikut di terminal terpisah untuk melampirkan standar input, output, dan kesalahan terminal Anda ke kontainer yang sedang berjalan. Hal ini memungkinkan Anda untuk melihat dan mengontrol kontainer Docker dari terminal Anda saat ini.

```
docker attach container-id
```

- Jalankan perintah berikut pada terminal yang terpisah. Hal ini memungkinkan Anda untuk menjalankan perintah Anda dalam mode interaktif, bahkan jika kontainer tidak terlampir.

```
docker exec -it container-id sh -c "command > /proc/1/fd/1"
```

Untuk pemecahan masalah AWS IoT Greengrass umum, lihat Pemecahan Masalah.

Konfigurasikan perangkat lunak AWS IoT Greengrass Inti

Perangkat lunak AWS IoT Greengrass Core menyediakan opsi yang dapat Anda gunakan untuk mengkonfigurasi perangkat lunak. Anda dapat membuat penerapan untuk mengonfigurasi perangkat lunak AWS IoT Greengrass Core pada setiap perangkat inti.

Topik

- Menyebarkan komponen inti Greengrass
- Konfigurasikan inti Greengrass sebagai layanan sistem
- Kontrol alokasi memori dengan opsi JVM
- Konfigurasikan pengguna yang menjalankan komponen
- Konfigurasikan batas sumber daya sistem untuk komponen
- Hubungkan pada port 443 atau melalui proksi jaringan
- Menggunakan sertifikat perangkat yang ditandatangani oleh CA pribadi
- Konfigurasikan pengaturan batas waktu dan cache MQTT

Menyebarkan komponen inti Greengrass

AWS IoT Greengrass menyediakan perangkat lunak AWS IoT Greengrass Core sebagai komponen yang dapat Anda terapkan ke perangkat inti Greengrass Anda. Anda dapat membuat deployment untuk menerapkan konfigurasi yang sama untuk beberapa perangkat inti Greengrass. Untuk

informasi selengkapnya, lihat Inti Greengrass dan Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Konfigurasikan inti Greengrass sebagai layanan sistem

Anda harus mengonfigurasi perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem di sistem init perangkat Anda untuk melakukan hal berikut:

- Mulai perangkat lunak AWS IoT Greengrass inti saat perangkat melakukan booting. Merupakan praktik yang baik jika Anda mengelola armada perangkat yang besar.
- Instal dan jalankan komponen plugin. Beberapa komponen AWS yang disediakan adalah komponen plugin, yang memungkinkannya untuk berinteraksi langsung dengan inti Greengrass. Untuk informasi selengkapnya tentang jenis komponen, lihatJenis komponen.
- Terapkan pembaruan over-the-air (OTA) ke perangkat lunak AWS IoT Greengrass Core perangkat inti. Untuk informasi selengkapnya, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).
- Aktifkan komponen untuk memulai ulang perangkat lunak Inti atau perangkat inti saat penerapan memperbarui komponen ke versi baru atau memperbarui parameter konfigurasi tertentu. AWS IoT Greengrass Untuk informasi lebih lanjut, lihat langkah siklus hidup bootstrap.

> **Important**
>
> Pada perangkat inti Windows, Anda harus mengatur perangkat lunak AWS IoT Greengrass inti sebagai layanan sistem.

Topik

- Konfigurasikan inti sebagai layanan sistem (Linux)
- Konfigurasikan nukleus sebagai layanan sistem (Windows)

Konfigurasikan inti sebagai layanan sistem (Linux)

Perangkat Linux mendukung sistem init yang berbeda, seperti initd, systemd, dan SystemV. Anda menggunakan `--setup-system-service true` argumen saat Anda menginstal perangkat lunak AWS IoT Greengrass Core untuk memulai inti sebagai layanan sistem dan mengonfigurasinya untuk diluncurkan saat perangkat melakukan booting. Installer mengkonfigurasi perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem dengan systemd.

Anda juga dapat mengonfigurasi inti secara manual untuk dijalankan sebagai layanan sistem. Contoh berikut adalah file layanan untuk systemd.

```
[Unit]
Description=Greengrass Core

[Service]
Type=simple
PIDFile=/greengrass/v2/alts/loader.pid
RemainAfterExit=no
Restart=on-failure
RestartSec=10
ExecStart=/bin/sh /greengrass/v2/alts/current/distro/bin/loader

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

Setelah Anda mengkonfigurasi layanan sistem, Anda dapat menjalankan perintah berikut untuk mengkonfigurasi memulai perangkat saat boot dan untuk memulai atau menghentikan perangkat lunak AWS IoT Greengrass Core.

- Untuk memeriksa status layanan (systemd)

  ```
  sudo systemctl status greengrass.service
  ```

- Untuk mengaktifkan inti untuk memulai saat perangkat melakukan booting.

  ```
  sudo systemctl enable greengrass.service
  ```

- Untuk menghentikan inti dari memulai saat perangkat melakukan booting.

  ```
  sudo systemctl disable greengrass.service
  ```

- Untuk memulai perangkat lunak AWS IoT Greengrass inti.

  ```
  sudo systemctl start greengrass.service
  ```

- Untuk menghentikan perangkat lunak AWS IoT Greengrass inti.

  ```
  sudo systemctl stop greengrass.service
  ```

Konfigurasikan nukleus sebagai layanan sistem (Windows)

Anda menggunakan `--setup-system-service true` argumen saat menginstal perangkat lunak AWS IoT Greengrass Core untuk memulai inti sebagai layanan Windows dan mengonfigurasinya untuk diluncurkan saat perangkat melakukan booting.

Setelah Anda mengkonfigurasi layanan, Anda dapat menjalankan perintah berikut untuk mengkonfigurasi memulai perangkat saat boot dan untuk memulai atau menghentikan perangkat lunak AWS IoT Greengrass Core. Anda harus menjalankan Command Prompt atau PowerShell sebagai administrator untuk menjalankan perintah ini.

Windows Command Prompt (CMD)

- Untuk memeriksa status layanan

  ```
  sc query "greengrass"
  ```

- Untuk mengaktifkan inti untuk memulai saat perangkat melakukan booting.

  ```
  sc config "greengrass" start=auto
  ```

- Untuk menghentikan inti dari memulai saat perangkat melakukan booting.

  ```
  sc config "greengrass" start=disabled
  ```

- Untuk memulai perangkat lunak AWS IoT Greengrass inti.

  ```
  sc start "greengrass"
  ```

- Untuk menghentikan perangkat lunak AWS IoT Greengrass inti.

  ```
  sc stop "greengrass"
  ```

  > **Note**
  >
  > Pada perangkat Windows, perangkat lunak AWS IoT Greengrass Core mengabaikan sinyal shutdown ini saat mematikan proses komponen Greengrass. Jika perangkat lunak AWS IoT Greengrass Core mengabaikan sinyal shutdown saat Anda menjalankan perintah ini, tunggu beberapa detik, dan coba lagi.

PowerShell

- Untuk memeriksa status layanan

```
Get-Service -Name "greengrass"
```

- Untuk mengaktifkan inti untuk memulai saat perangkat melakukan booting.

```
Set-Service -Name "greengrass" -Status stopped -StartupType automatic
```

- Untuk menghentikan inti dari memulai saat perangkat melakukan booting.

```
Set-Service -Name "greengrass" -Status stopped -StartupType disabled
```

- Untuk memulai perangkat lunak AWS IoT Greengrass inti.

```
Start-Service -Name "greengrass"
```

- Untuk menghentikan perangkat lunak AWS IoT Greengrass inti.

```
Stop-Service -Name "greengrass"
```

> **Note**
>
> Pada perangkat Windows, perangkat lunak AWS IoT Greengrass Core mengabaikan sinyal shutdown ini saat mematikan proses komponen Greengrass. Jika perangkat lunak AWS IoT Greengrass Core mengabaikan sinyal shutdown saat Anda menjalankan perintah ini, tunggu beberapa detik, dan coba lagi.

Kontrol alokasi memori dengan opsi JVM

Jika Anda menjalankan AWS IoT Greengrass perangkat dengan memori terbatas, Anda dapat menggunakan opsi mesin virtual Java (JVM) untuk mengontrol ukuran tumpukan maksimum, mode pengumpulan sampah, dan opsi kompiler, yang mengontrol jumlah memori yang AWS IoT Greengrass digunakan perangkat lunak Core. Ukuran tumpukan di JVM menentukan berapa banyak memori yang dapat digunakan aplikasi sebelum pengumpulan sampah terjadi, atau sebelum aplikasi kehabisan memori. Ukuran tumpukan maksimum menentukan jumlah maksimum memori yang dapat dialokasikan oleh JVM ketika memperluas tumpukan selama aktivitas yang berat.

Untuk mengontrol alokasi memori, buat deployment baru atau revisi deployment yang ada yang mencakup komponen inti, dan tentukan opsi JVM Anda di parameter konfigurasi `jvmOptions` dalam konfigurasi komponen inti.

Bergantung pada kebutuhan Anda, Anda dapat menjalankan perangkat lunak AWS IoT Greengrass Core dengan alokasi memori yang dikurangi atau dengan alokasi memori minimum.

Alokasi memori yang berkurang

Untuk menjalankan perangkat lunak AWS IoT Greengrass Core dengan alokasi memori yang dikurangi, kami menyarankan Anda menggunakan contoh pembaruan gabungan konfigurasi berikut untuk menyetel opsi JVM dalam konfigurasi inti Anda:

```
{
  "jvmOptions": "-Xmx64m -XX:+UseSerialGC -XX:TieredStopAtLevel=1"
}
```

Alokasi memori minimum

Untuk menjalankan perangkat lunak AWS IoT Greengrass Core dengan alokasi memori minimum, kami menyarankan Anda menggunakan contoh pembaruan gabungan konfigurasi berikut untuk menyetel opsi JVM dalam konfigurasi inti Anda:

```
{
  "jvmOptions": "-Xmx32m -XX:+UseSerialGC -Xint"
}
```

Konfigurasi ini menggabungkan pembaruan dengan menggunakan opsi JVM berikut:

`-Xmx`*NN*`m`

Menetapkan ukuran tumpukan JVM maksimum.

Untuk alokasi memori yang berkurang, gunakan `-Xmx64m` sebagai nilai awal untuk membatasi ukuran tumpukan 64 MB. Untuk alokasi memori minimum, gunakan `-Xmx32m` sebagai nilai awal untuk membatasi ukuran tumpukan 32 MB.

Anda dapat menambah atau mengurangi nilai `-Xmx` tergantung pada kebutuhan aktual Anda; namun, kami sangat menyarankan Anda tidak menetapkan ukuran tumpukan maksimum di bawah 16 MB. Jika ukuran heap maksimum terlalu rendah untuk lingkungan Anda, maka

perangkat lunak AWS IoT Greengrass Core mungkin mengalami kesalahan yang tidak terduga karena memori yang tidak mencukupi.

`-XX:+UseSerialGC`

Menentukan untuk menggunakan koleksi sampah serial untuk ruang tumpukan JVM. Kolektor sampah serial lebih lambat, tetapi menggunakan memori lebih kecil dari implementasi pengumpulan sampah JVM lainnya.

`-XX:TieredStopAtLevel=1`

Menginstruksikan JVM untuk menggunakan compiler Java just-in-time (JIT) sekali. Karena kode yang dikompilasi JIT menggunakan ruang dalam memori perangkat, penggunaan compiler JIT lebih dari sekali akan mengonsumsi lebih banyak memori daripada kompilasi tunggal.

`-Xint`

Menginstruksikan JVM untuk tidak menggunakan just-in-time (JIT) compiler. Sebaliknya, JVM berjalan dalam mode ditafsirkan-saja. Mode ini lebih lambat daripada menjalankan kode yang dikompilasi JIT; namun, kode yang dikompilasi itu tidak menggunakan ruang apa pun dalam memori.

Untuk informasi tentang pembuatan pembaruan penggabungan konfigurasi, lihat Perbarui konfigurasi komponen.

Konfigurasikan pengguna yang menjalankan komponen

Perangkat lunak AWS IoT Greengrass Core dapat menjalankan proses komponen sebagai pengguna sistem dan kelompok yang berbeda dari yang menjalankan perangkat lunak. Ini meningkatkan keamanan, karena Anda dapat menjalankan perangkat lunak AWS IoT Greengrass Core sebagai root, atau sebagai pengguna administrator, tanpa memberikan izin tersebut ke komponen yang berjalan pada perangkat inti.

Tabel berikut menunjukkan jenis komponen mana yang dapat dijalankan oleh perangkat lunak AWS IoT Greengrass Core sebagai pengguna yang Anda tentukan. Untuk informasi selengkapnya, lihat Jenis komponen.

Jenis komponen	Konfigurasikan pengguna komponen
Inti	 Tidak
Plugin	 Tidak
Generik	 Ya
Lambda (tidak terkontainerisasi)	 Ya
Lambda (terkontainerisasi)	 Ya

Anda harus membuat pengguna komponen sebelum Anda dapat menentukannya dalam konfigurasi penerapan. Pada perangkat berbasis Windows, Anda juga harus menyimpan nama pengguna dan kata sandi untuk pengguna di instance pengelola kredensi akun. LocalSystem Untuk informasi selengkapnya, lihat Siapkan pengguna komponen di perangkat Windows.

Saat mengonfigurasi pengguna komponen pada perangkat berbasis Linux, Anda juga dapat menentukan grup secara opsional. Anda menentukan pengguna dan grup yang dipisahkan oleh titik dua (`:`) dalam format berikut:*`user`*`:`*`group`*. Jika Anda tidak menentukan grup, perangkat lunak AWS

IoT Greengrass Core default ke grup utama pengguna. Anda dapat menggunakan nama atau ID untuk mengidentifikasi pengguna dan grup.

Pada perangkat berbasis Linux, Anda juga dapat menjalankan komponen sebagai pengguna sistem yang tidak ada, juga disebut pengguna yang tidak dikenal, untuk meningkatkan keamanan. Proses Linux dapat menandakan proses lain yang dijalankan oleh pengguna yang sama. Pengguna yang tidak dikenal tidak menjalankan proses yang lain, sehingga Anda dapat menjalankan komponen sebagai pengguna yang tidak dikenal untuk mencegah komponen memberi sinyal ke komponen lain pada perangkat inti. Untuk menjalankan komponen sebagai pengguna yang tidak dikenal, tentukan ID pengguna yang tidak ada pada perangkat inti. Anda juga dapat menentukan ID grup yang tidak ada untuk dijalankan sebagai grup yang tidak dikenal.

Anda dapat mengkonfigurasi pengguna untuk setiap komponen dan untuk setiap perangkat inti.

- Konfigurasikan untuk komponen

  Anda dapat mengonfigurasi setiap komponen untuk dijalankan dengan pengguna khusus untuk komponen tersebut. Saat membuat penerapan, Anda dapat menentukan pengguna untuk setiap komponen dalam `runWith` konfigurasi untuk komponen tersebut. Perangkat lunak AWS IoT Greengrass Core menjalankan komponen sebagai pengguna yang ditentukan jika Anda mengonfigurasinya. Jika tidak, default menjalankan komponen sebagai pengguna default yang Anda konfigurasikan untuk perangkat inti. Untuk informasi selengkapnya tentang menentukan pengguna komponen dalam konfigurasi penerapan, lihat parameter `runWith`konfigurasi di. Buat deployment
- Konfigurasikan pengguna default untuk perangkat inti

  Anda dapat mengonfigurasi pengguna default yang digunakan perangkat lunak AWS IoT Greengrass Core untuk menjalankan komponen. Ketika perangkat lunak AWS IoT Greengrass Core menjalankan komponen, ia memeriksa apakah Anda menentukan pengguna untuk komponen itu, dan menggunakannya untuk menjalankan komponen. Jika komponen tidak menentukan pengguna, maka perangkat lunak AWS IoT Greengrass Core menjalankan komponen sebagai pengguna default yang Anda konfigurasikan untuk perangkat inti. Untuk informasi selengkapnya, lihat Konfigurasikan pengguna komponen default.

> **Note**
>
> Pada perangkat berbasis Windows, Anda harus menentukan setidaknya pengguna default untuk menjalankan komponen.

Pada perangkat berbasis Linux, pertimbangan berikut berlaku jika Anda tidak mengonfigurasi pengguna untuk menjalankan komponen:

- Jika Anda menjalankan perangkat lunak AWS IoT Greengrass Core sebagai root, maka perangkat lunak tidak akan menjalankan komponen. Anda harus menentukan pengguna default untuk menjalankan komponen jika Anda berjalan sebagai root.
- Jika Anda menjalankan perangkat lunak AWS IoT Greengrass Core sebagai pengguna non-root, maka perangkat lunak menjalankan komponen sebagai pengguna tersebut.

Topik

Siapkan pengguna komponen di perangkat Windows

Untuk mengatur pengguna komponen pada perangkat berbasis Windows

1. Buat pengguna komponen di LocalSystem akun di perangkat.

   ```
   net user /add component-user password
   ```

2. Gunakan PsExec utilitas Microsoft untuk menyimpan nama pengguna dan kata sandi untuk pengguna komponen dalam contoh Credential Manager untuk LocalSystem akun tersebut.

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

   > **Note**
   >
   > Pada perangkat berbasis Windows, LocalSystem akun menjalankan inti Greengrass, dan Anda harus menggunakan PsExec utilitas untuk menyimpan informasi pengguna komponen di akun. LocalSystem Menggunakan aplikasi Credential Manager menyimpan informasi ini di akun Windows dari pengguna yang saat ini masuk, bukan LocalSystem akun.

Konfigurasikan pengguna komponen default

Anda dapat menggunakan penerapan untuk mengonfigurasi pengguna default pada perangkat inti. Dalam deployment ini, Anda akan memperbarui konfigurasi komponen inti.

> **Note**
>
> Anda juga dapat mengatur pengguna default ketika Anda menginstal perangkat lunak AWS IoT Greengrass Core dengan `--component-default-user` opsi. Untuk informasi selengkapnya, lihat Instal perangkat lunak inti AWS IoT Greengrass.

Buat penyebaran yang menentukan pemutakhiran konfigurasi berikut untuk komponen. `aws.greengrass.Nucleus`

Linux

```
{
  "runWithDefault": {
    "posixUser": "ggc_user:ggc_group"
  }
}
```

Windows

```
{
  "runWithDefault": {
    "windowsUser": "ggc_user"
  }
}
```

> **Note**
>
> Pengguna yang Anda tentukan harus ada, dan nama pengguna dan kata sandi untuk pengguna ini harus disimpan dalam contoh pengelola kredensi LocalSystem akun di perangkat Windows Anda. Untuk informasi selengkapnya, lihat Siapkan pengguna komponen di perangkat Windows.

Contoh berikut mendefinisikan penerapan untuk perangkat berbasis Linux yang mengkonfigurasi `ggc_user` sebagai pengguna default dan sebagai grup default. `ggc_group` Pembaruan konfigurasi `merge` memerlukan objek JSON berserial.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.6",
      "configurationUpdate": {
        "merge": "{\"runWithDefault\":{\"posixUser\":\"ggc_user:ggc_group\"}}"
      }
    }
  }
}
```

Konfigurasikan batas sumber daya sistem untuk komponen

> **Note**
>
> Fitur ini tersedia untuk v2.4.0 dan yang lebih baru dari komponen inti Greengrass. AWS IoT Greengrass saat ini tidak mendukung fitur ini di perangkat inti Windows.

Anda dapat mengonfigurasi jumlah maksimum penggunaan CPU dan RAM yang dapat digunakan oleh setiap proses komponen pada perangkat inti.

Tabel berikut menunjukkan jenis komponen yang mendukung batas sumber daya sistem. Untuk informasi selengkapnya, lihat Jenis komponen.

Jenis komponen	Konfigurasikan batas sumber daya sistem
Inti	 Tidak

Jenis komponen	Konfigurasikan batas sumber daya sistem
Plugin	 Tidak
Generik	 Ya
Lambda (tidak terkontainerisasi)	 Ya
Lambda (terkontainerisasi)	 Tidak

> **Important**
>
> Batas sumber daya sistem tidak didukung saat Anda menjalankan perangkat lunak AWS IoT Greengrass Core dalam wadah Docker.

Anda dapat mengonfigurasi batas sumber daya sistem untuk setiap komponen dan untuk setiap perangkat inti.

- Konfigurasikan untuk komponen

  Anda dapat mengonfigurasi setiap komponen dengan batas sumber daya sistem khusus untuk komponen tersebut. Saat membuat penerapan, Anda dapat menentukan batas sumber daya sistem untuk setiap komponen dalam penerapan. Jika komponen mendukung batas sumber daya sistem, perangkat lunak AWS IoT Greengrass Core menerapkan batasan untuk proses komponen.

Jika Anda tidak menentukan batas sumber daya sistem untuk suatu komponen, perangkat lunak AWS IoT Greengrass Core menggunakan default apa pun yang telah Anda konfigurasikan untuk perangkat inti. Untuk informasi selengkapnya, lihat Buat deployment.

- Konfigurasikan default untuk perangkat inti

  Anda dapat mengonfigurasi batas sumber daya sistem default yang diterapkan perangkat lunak AWS IoT Greengrass Core untuk komponen yang mendukung batasan ini. Ketika perangkat lunak AWS IoT Greengrass Core menjalankan komponen, itu menerapkan batas sumber daya sistem yang Anda tentukan untuk komponen itu. Jika komponen tersebut tidak menentukan batas sumber daya sistem, perangkat lunak AWS IoT Greengrass Core menerapkan batas sumber daya sistem default yang Anda konfigurasikan untuk perangkat inti. Jika Anda tidak menentukan batas sumber daya sistem default, perangkat lunak AWS IoT Greengrass Core tidak menerapkan batasan sumber daya sistem apa pun secara default. Untuk informasi selengkapnya, lihat Konfigurasikan batas sumber daya sistem default.

Konfigurasikan batas sumber daya sistem default

Anda dapat menerapkan komponen inti Greengrass untuk mengonfigurasi batas sumber daya sistem default untuk perangkat inti. Untuk mengonfigurasi batas sumber daya sistem default, buat penerapan yang menentukan pemutakhiran konfigurasi berikut untuk komponen tersebut`aws.greengrass.Nucleus`.

```
{
  "runWithDefault": {
    "systemResourceLimits": {
      "cpu": cpuTimeLimit,
      "memory": memoryLimitInKb
    }
  }
}
```

Contoh berikut mendefinisikan penerapan yang mengkonfigurasi batas waktu CPU2, yang setara dengan penggunaan 50% pada perangkat dengan 4 core CPU. Contoh ini juga mengkonfigurasi penggunaan memori hingga 100 MB.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.6",
```

```
      "configurationUpdate": {
        "merge": "{\"runWithDefault\":{\"systemResourceLimits\":\"cpus\":2,\"memory
\":102400}}}"
      }
    }
  }
}
```

Hubungkan pada port 443 atau melalui proksi jaringan

AWS IoT Greengrass perangkat inti berkomunikasi dengan AWS IoT Core menggunakan protokol pesan MQTT dengan otentikasi klien TLS. Menurut 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 restriktif lainnya mungkin memerlukan semua lalu lintas untuk melalui proxy sebelum terhubung ke internet.

> **Note**
>
> Perangkat inti Greengrass yang menjalankan komponen inti Greengrass v2.0.3 dan sebelumnya menggunakan port 8443 untuk terhubung ke titik akhir bidang data. AWS IoT Greengrass Perangkat ini harus dapat terhubung ke titik akhir ini pada port 8443. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

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

- Komunikasi MQTT melalui port 443. Jika jaringan Anda memungkinkan koneksi ke port 443, Anda dapat mengonfigurasi perangkat inti Greengrass dengan menggunakan port 443 untuk lalu lintas MQTT, dan bukan port default 8883. Hal ini bisa menjadi koneksi langsung ke port 443 atau koneksi melalui server proksi jaringan. Tidak seperti konfigurasi default, yang menggunakan autentikasi klien berbasis sertifikat, MQTT pada port 443 menggunakan peran layanan perangkat untuk autentikasi.

  Untuk informasi selengkapnya, lihat Konfigurasikan MQTT melalui port 443.
- Komunikasi HTTPS melalui port 443. Perangkat lunak AWS IoT Greengrass Core mengirimkan lalu lintas HTTPS melalui port 8443 secara default, tetapi Anda dapat mengonfigurasinya untuk

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

> **Important**
>
> Untuk menggunakan ALPN dan mengaktifkan komunikasi HTTPS melalui port 443, perangkat inti Anda harus menjalankan Java 8 update 252 atau yang lebih baru. Semua pembaruan Java versi 9 dan yang lebih baru juga mendukung ALPN.

Untuk informasi selengkapnya, lihat Konfigurasikan HTTPS melalui port 443.

- Sambungan melalui proksi rangkaian. Anda dapat mengonfigurasi server proxy jaringan untuk bertindak sebagai perantara untuk menghubungkan ke perangkat inti Greengrass. AWS IoT Greengrass mendukung otentikasi dasar untuk proxy HTTP dan HTTPS.

  Perangkat inti Greengrass harus menjalankan Greengrass nucleus v2.5.0 atau yang lebih baru untuk menggunakan proxy HTTPS.

  Perangkat lunak AWS IoT Greengrass Core meneruskan konfigurasi proxy ke komponen melaluiALL_PROXY,, HTTP_PROXYHTTPS_PROXY, dan variabel NO_PROXY lingkungan. Komponen harus menggunakan pengaturan ini untuk terhubung melalui proksi. Komponen menggunakan pustaka umum (seperti boto3, cURL, dan paket python `requests`) yang biasanya menggunakan variabel lingkungan ini secara default untuk membuat koneksi. Jika suatu komponen juga menentukan variabel lingkungan ini, AWS IoT Greengrass tidak akan menimpanya.

  Untuk informasi selengkapnya, lihat Konfigurasikan proksi jaringan.

Konfigurasikan MQTT melalui port 443

Anda dapat mengonfigurasi MQTT melalui port 443 pada perangkat inti yang ada atau saat Anda menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti baru.

Topik

- Konfigurasikan MQTT melalui port 443 pada perangkat inti yang ada
- Konfigurasikan MQTT melalui port 443 selama instalasi

Konfigurasikan MQTT melalui port 443 pada perangkat inti yang ada

Anda dapat menggunakan deployment untuk mengonfigurasi MQTT melalui port 443 pada perangkat inti tunggal atau sekelompok perangkat inti. Dalam deployment ini, Anda akan memperbarui konfigurasi komponen inti. Nukleus akan dimulai ulang ketika Anda memperbarui konfigurasi `mqtt`-nya.

Untuk mengonfigurasi MQTT melalui port 443, buat deployment yang menentukan pembaruan konfigurasi berikut untuk komponen `aws.greengrass.Nucleus`.

```
{
  "mqtt": {
    "port": 443
  }
}
```

Contoh berikut menentukan deployment yang mengonfigurasi MQTT melalui port 443. Pembaruan konfigurasi `merge` memerlukan objek JSON berserial.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.6",
      "configurationUpdate": {
        "merge": "{\"mqtt\":{\"port\":443}}"
      }
    }
  }
}
```

Konfigurasikan MQTT melalui port 443 selama instalasi

Anda dapat mengonfigurasi MQTT melalui port 443 saat Anda menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti. Gunakan argumen `--init-config` installer untuk mengkonfigurasi MQTT melalui port 443. Anda dapat menentukan argumen ini saat menginstal dengan penyediaan manual, penyediaanarmada, atau penyediaan kustom.

Konfigurasikan HTTPS melalui port 443

Fitur ini memerlukan v2.0.4 Inti Greengrass atau lebih baru.

Anda dapat mengonfigurasi HTTPS melalui port 443 pada perangkat inti yang ada atau saat Anda menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti baru.

Topik

Konfigurasikan HTTPS melalui port 443 pada perangkat inti yang ada

Anda dapat menggunakan deployment untuk mengonfigurasi HTTPS melalui port 443 pada perangkat inti tunggal atau sekelompok perangkat inti. Dalam deployment ini, Anda akan memperbarui konfigurasi komponen inti.

Untuk mengonfigurasi HTTPS melalui port 443, buat deployment yang menentukan pembaruan konfigurasi berikut untuk komponen `aws.greengrass.Nucleus`.

```
{
  "greengrassDataPlanePort": 443
}
```

Contoh berikut menentukan deployment yang mengonfigurasi HTTPS melalui port 443. Pembaruan konfigurasi `merge` memerlukan objek JSON berserial.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.6",
      "configurationUpdate": {
        "merge": "{\"greengrassDataPlanePort\":443}"
      }
    }
  }
}
```

Konfigurasikan HTTPS melalui port 443 selama instalasi

Anda dapat mengonfigurasi HTTPS melalui port 443 saat Anda menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti. Gunakan argumen `--init-config` installer untuk mengonfigurasi HTTPS melalui port 443. Anda dapat menentukan argumen ini saat menginstal dengan penyediaan manual, penyediaanarmada, atau penyediaan kustom.

Konfigurasikan proksi jaringan

Ikuti prosedur di bagian ini untuk mengonfigurasi perangkat inti Greengrass untuk terhubung ke internet melalui proxy jaringan HTTP atau HTTPS. Untuk informasi selengkapnya tentang titik akhir dan port yang digunakan perangkat inti, lihatIzinkan lalu lintas perangkat melalui proxy atau firewall.

Important

Jika perangkat inti Anda menjalankan versi inti Greengrass lebih awal dari v2.4.0, peran perangkat Anda harus mengizinkan izin berikut untuk menggunakan proxy jaringan:

- `iot:Connect`
- `iot:Publish`
- `iot:Receive`
- `iot:Subscribe`

Ini diperlukan karena perangkat menggunakan AWS kredensional dari layanan pertukaran token untuk mengautentikasi koneksi MQTT ke. AWS IoT Perangkat menggunakan MQTT untuk menerima dan menginstal penerapan dari perangkat AWS Cloud, sehingga perangkat Anda tidak akan berfungsi kecuali Anda menentukan izin ini pada perannya. Perangkat biasanya menggunakan sertifikat X.509 untuk mengautentikasi koneksi MQTT, tetapi perangkat tidak dapat melakukan hal ini untuk mengautentikasi saat menggunakan proksi. Untuk informasi lebih lanjut tentang cara mengonfigurasi peran perangkat, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Topik

Konfigurasikan proxy jaringan pada perangkat inti yang ada

Anda dapat menggunakan deployment untuk mengonfigurasi proksi jaringan pada perangkat inti tunggal atau sekelompok perangkat inti. Dalam deployment ini, Anda akan memperbarui konfigurasi

komponen inti. Nukleus akan dimulai ulang ketika Anda memperbarui konfigurasi `networkProxy`-nya.

Untuk mengonfigurasi proksi jaringan, buat deployment untuk komponen `aws.greengrass.Nucleus` yang menggabungkan pembaruan konfigurasi berikut. Pembaruan konfigurasi ini berisi objek networkProxy.

```
{
  "networkProxy": {
    "noProxyAddresses": "http://192.168.0.1,www.example.com",
    "proxy": {
      "url": "https://my-proxy-server:1100"
    }
  }
}
```

Contoh berikut menentukan deployment yang mengonfigurasi proksi jaringan. Pembaruan konfigurasi `merge` memerlukan objek JSON berserial.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.6",
      "configurationUpdate": {
        "merge": "{\"networkProxy\":{\"noProxyAddresses\":
\"http://192.168.0.1,www.example.com\",\"proxy\":{\"url\":\"https://my-proxy-
server:1100\",\"username\":\"Mary_Major\",\"password\":\"pass@word1357\"}}}"
      }
    }
  }
}
```

Konfigurasikan proxy jaringan selama instalasi

Anda dapat mengkonfigurasi proxy jaringan ketika Anda menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti. Gunakan argumen `--init-config` installer untuk mengkonfigurasi proxy jaringan. Anda dapat menentukan argumen ini saat menginstal dengan penyediaan manual, penyediaanarmada, atau penyediaan kustom.

Aktifkan perangkat inti untuk mempercayai proxy HTTPS

Saat Anda mengonfigurasi perangkat inti untuk menggunakan proxy HTTPS, Anda harus menambahkan rantai sertifikat server proxy ke perangkat inti untuk memungkinkannya mempercayai proxy HTTPS. Jika tidak, perangkat inti mungkin mengalami kesalahan saat mencoba merutekan lalu lintas melalui proxy. Tambahkan sertifikat CA server proxy ke file sertifikat CA root Amazon perangkat inti.

Untuk mengaktifkan perangkat inti untuk mempercayai proxy HTTPS

1. Temukan file sertifikat CA root Amazon di perangkat inti.

   - Jika Anda menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan otomatis, file sertifikat CA root Amazon ada di. */greengrass/v2*`/rootCA.pem`
   - Jika Anda menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan manual atau armada, file sertifikat CA root Amazon mungkin ada di. */greengrass/v2*`/AmazonRootCA1.pem`

   Jika sertifikat CA root Amazon tidak ada di lokasi ini, periksa `system.rootCaPath` properti */greengrass/v2*`/config/effectiveConfig.yaml` untuk menemukan lokasinya.
2. Tambahkan konten file sertifikat CA server proxy ke file sertifikat CA root Amazon.

   Contoh berikut menunjukkan sertifikat CA server proxy yang ditambahkan ke file sertifikat CA root Amazon.

```
-----BEGIN CERTIFICATE-----
MIIEFTCCAv2gAwIQWgIVAMHSAzWG/5YVRYtRQOxXUTEpHuEmApzGCSqGSIb3DQEK
\nCwUAhuL9MQswCQwJVUzEPMAVUzEYMBYGA1UECgwP1hem9uLmNvbSBJbmMuMRww
... content of proxy CA certificate ...
+vHIRlt0e5JAm5\noTIZGoFbK82A0/nO7f/t5PSIDAim9V3Gc3pSXxCCAQoFYnui
GaPUlGk1gCE84a0X\n7Rp/lND/PuMZ/s8YjlkY2NmYmNjMCAXDTE5MTEyN2cM216
gJMIADggEPADf2/m45hzEXAMPLE=
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIDQTCCAimgF6AwIBAgITBmyfz/5mjAo54vB4ikPmljZKyjANJmApzyMZFo6qBg
ADA5MQswCQYDVQQGEwJVUzEPMA0tMVT8QtPHRh8jrdkGA1UEChMGDV3QQDExBBKW
... content of root CA certificate ...
o/ufQJQWUCyziar1hem9uMRkwFwYVPSHCb2XV4cdFyQzR1KldZwgJcIQ6XUDgHaa
5MsI+yMRQ+hDaXJiobldXgjUka642M4UwtBV8oK2xJNDd2ZhwLnoQdeXeGADKkpy
```

```
rqXRfKoQnoZsG4q5WTP46EXAMPLE
-----END CERTIFICATE-----
```

Objek networkProxy

Gunakan objek `networkProxy` untuk menentukan informasi tentang proksi jaringan. Objek ini berisi informasi berikut:

`noProxyAddresses`

(Opsional) Daftar alamat IP atau nama host yang dipisahkan dengan koma yang dikecualikan dari proksi.

`proxy`

Proksi yang akan dihubungkan. Objek ini berisi informasi berikut:

`url`

URL server proksi dalam format `scheme://userinfo@host:port`.

- `scheme` — Skema, yang harus berupa `http` atau `https`.

  > **Important**
  >
  > Perangkat inti Greengrass harus menjalankan Greengrass nucleus v2.5.0 atau yang lebih baru untuk menggunakan proxy HTTPS.
  > Jika Anda mengonfigurasi proxy HTTPS, Anda harus menambahkan sertifikat CA server proxy ke sertifikat CA root Amazon perangkat inti. Untuk informasi selengkapnya, lihat Aktifkan perangkat inti untuk mempercayai proxy HTTPS.

- `userinfo` - (Opsional) Nama pengguna dan informasi kata sandi. Jika Anda menentukan informasi ini di`url`, perangkat inti Greengrass mengabaikan bidang dan. `username` `password`
- `host` - Nama host atau alamat IP server proksi.
- `port` — (Opsional) Nomor port. Jika Anda tidak menentukan port, maka perangkat inti Greengrass akan menggunakan nilai default berikut:
  - `http` – 80
  - `https` – 443

`username`

(Opsional) Nama pengguna yang mengautentikasi server proxy.

`password`

(Opsional) Kata sandi yang mengautentikasi server proxy.

Menggunakan sertifikat perangkat yang ditandatangani oleh CA pribadi

Jika Anda menggunakan otoritas sertifikat pribadi kustom (CA), Anda harus menyetel inti Greengrass ke. **`greengrassDataPlaneEndpoint iotdata`** Anda dapat mengatur opsi ini selama penerapan atau instalasi menggunakan argumen **`--init-config`** installer.

Anda dapat menyesuaikan titik akhir bidang data Greengrass tempat perangkat terhubung. Anda dapat mengatur opsi konfigurasi ini **`iotdata`** untuk mengatur titik akhir bidang data Greengrass ke titik akhir yang sama dengan titik akhir data IoT, yang dapat Anda tentukan dengan. **`iotDataEndpoint`**

Konfigurasikan pengaturan batas waktu dan cache MQTT

Di AWS IoT Greengrass lingkungan, komponen dapat menggunakan MQTT untuk berkomunikasi dengan. AWS IoT Core Perangkat lunak AWS IoT Greengrass Core mengelola pesan MQTT untuk komponen. Ketika perangkat inti kehilangan koneksi ke AWS Cloud, perangkat lunak akan menangkap pesan MQTT untuk mencoba lagi nanti ketika sambungan pulih kembali. Anda dapat mengonfigurasi pengaturan seperti timeout pesan dan ukuran cache. Untuk informasi lebih lanjut, lihat parameter konfigurasi `mqtt` dan `mqtt.spooler` pada komponen nukleus Greengrass.

AWS IoT Core memberlakukan kuota layanan pada broker pesan MQTT. Kuota ini mungkin berlaku untuk pesan yang Anda kirim antara perangkat inti dan AWS IoT Core. Untuk informasi lebih lanjut, lihat kuota layanan broker AWS IoT Core pesan di. Referensi Umum AWS

Perbarui perangkat lunak inti AWS IoT Greengrass (OTA)

Perangkat lunak AWS IoT Greengrass Core terdiri dari komponen inti Greengrass dan komponen opsional lainnya yang dapat Anda terapkan ke perangkat Anda untuk over-the-air melakukan pembaruan (OTA) perangkat lunak. Fitur ini dibangun pada perangkat lunak inti AWS IoT Greengrass.

Pembaruan OTA membuatnya lebih efisien untuk:

- Memperbaiki kerentanan keamanan.
- Mengatasi masalah stabilitas perangkat lunak.
- Men-deploy fitur baru atau yang lebih baik.

Topik

Persyaratan

Persyaratan berikut berlaku untuk men-deploy pembaruan OTA perangkat lunak inti AWS IoT Greengrass:

- Perangkat inti Greengrass harus memiliki koneksi ke AWS Cloud untuk menerima deployment.
- Perangkat inti Greengrass harus dikonfigurasi dengan benar dan ditetapkan dengan sertifikat dan kunci untuk autentikasi dengan AWS IoT Core dan AWS IoT Greengrass.
- Perangkat lunak inti AWS IoT Greengrass harus disiapkan dan dijalankan sebagai layanan sistem. Pembaruan OTA tidak bekerja jika Anda menjalankan inti dari file JAR, `Greengrass.jar`. Untuk informasi selengkapnya, lihat Konfigurasikan inti Greengrass sebagai layanan sistem.

Pertimbangan untuk perangkat inti

Sebelum Anda melakukan pembaruan OTA, perhatikan dampaknya pada perangkat inti yang Anda perbarui dan perangkat kliennya yang terhubung:

- Nukleus Greengrass dimatikan.
- Semua komponen yang berjalan pada perangkat inti juga dimatikan. Jika komponen tersebut menuliskan ke sumber daya lokal, komponen itu mungkin meninggalkan sumber daya tersebut dalam keadaan salah kecuali jika dimatikan dengan benar. Komponen dapat menggunakan komunikasi antarproses untuk memberitahu komponen inti untuk menunda pembaruan sampai komponen itu membersihkan sumber daya yang digunakannya.

- Sementara komponen nukleus dimatikan, perangkat inti kehilangan koneksi dengan AWS Cloud dan perangkat lokal. Perangkat inti tidak akan merutekan pesan dari perangkat klien saat dimatikan.
- Fungsi Lambda berusia panjang yang berjalan sebagai komponen akan kehilangan informasi keadaan dinamisnya dan menurunkan semua pekerjaan tertunda.

Perilaku pembaruan inti Greengrass

Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda dimulai ulang secara tiba-tiba.

Ketika versi komponen nukleus Greengrass berubah, perangkat lunak inti AWS IoT Greengrass—yang mencakup inti dan semua komponen lainnya di perangkat Anda—akan dimulai ulang untuk menerapkan perubahan itu. Karena dampak pada perangkat inti saat komponen nukleus diperbarui, Anda mungkin ingin mengontrol saat versi patch nukleus baru di-deploy ke perangkat Anda. Untuk melakukannya, Anda harus langsung menyertakan komponen inti Greengrass dalam deployment Anda. Menyertakan komponen secara langsung berarti Anda menyertakan versi tertentu dari komponen tersebut dalam konfigurasi deployment Anda dan tidak bergantung pada dependensi komponen untuk men-deploy komponen tersebut ke perangkat Anda. Untuk informasi lebih lanjut tentang penentuan dependensi dalam resep komponen Anda, lihat Format resep.

Tinjau tabel berikut untuk memahami perilaku pembaruan untuk komponen inti Greengrass berdasarkan tindakan dan deployment konfigurasi Anda.

Tindakan	Konfigurasi deployment	Perilaku pembaruan nukleus
Tambahkan perangkat baru ke grup objek yang ditargetkan oleh deployment yang ada tanpa merevisi deployment.	Deployment tidak secara langsung menyertakan nukleus Greengrass. Deployment langsung mencakup setidaknya satu komponen yang disediakan	Pada perangkat baru, instal versi patch terbaru dari nukleus yang memenuhi semua persyaratan dependensi komponen.

Tindakan	Konfigurasi deployment	Perilaku pembaruan nukleus
	oleh AWS, atau mencakup komponen kustom yang tergantung pada komponen yang disediakan oleh AWS atau pada nukleus Greengras s.	Pada perangkat yang ada, jangan perbarui versi nukleus yang sudah diinstal.
Tambahkan perangkat baru ke grup objek yang ditargetk an oleh deployment yang ada tanpa merevisi deployment.	Deployment secara langsung mencakup versi tertentu dari inti Greengrass.	Pada perangkat baru, instal versi nukleus yang ditentukan. Pada perangkat yang ada, jangan perbarui versi nukleus yang sudah diinstal.
Buat deployment baru atau revisi deployment yang ada.	Deployment tidak secara langsung menyertakan nukleus Greengrass. Deployment tersebut secara langsung mencakup setidakny a satu komponen yang disediakan oleh AWS, atau mencakup komponen kustom yang tergantung pada komponen yang disediakan oleh AWS atau pada nukleus Greengrass.	Pada semua perangkat yang ditargetkan, instal versi patch terbaru dari inti yang memenuhi semua persyarat an dependensi komponen, termasuk pada perangkat baru yang Anda tambahkan ke grup objek yang ditargetkan.

Tindakan	Konfigurasi deployment	Perilaku pembaruan nukleus
Buat deployment baru atau revisi deployment yang ada.	Deployment secara langsung mencakup versi tertentu dari inti Greengrass.	Pada semua perangkat yang ditargetkan, instal versi nukleus yang ditentukan, termasuk perangkat baru yang Anda tambahkan ke grup objek yang ditargetkan.

Lakukan pembaruan OTA

Untuk melakukan pembaruan OTA, buat deployment yang mencakup atribut komponen nukleus dan versi yang akan diinstal.

Hapus instalasi perangkat lunak inti AWS IoT Greengrass

Anda dapat menghapus instalasi perangkat lunak inti AWS IoT Greengrass untuk menghapusnya dari perangkat yang tidak ingin Anda gunakan sebagai perangkat inti Greengrass. Anda juga dapat menggunakan langkah-langkah ini untuk membersihkan instalasi yang gagal.

Untuk menghapus instalasi perangkat lunak inti AWS IoT Greengrass

1. Jika Anda menjalankan perangkat lunak sebagai layanan sistem, Anda harus menghentikan, menonaktifkan, dan menghapus layanan. Jalankan perintah berikut yang sesuai untuk sistem operasi Anda.

   Linux

   1. Hentikan layanan .

      ```
      sudo systemctl stop greengrass.service
      ```

   2. Nonaktifkan layanan.

      ```
      sudo systemctl disable greengrass.service
      ```

   3. Hapus layanan.

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

4. Verifikasi bahwa layanan dihapus.

```
sudo systemctl daemon-reload && sudo systemctl reset-failed
```

Windows (Command Prompt)

> **Note**
>
> Anda harus menjalankan Command Prompt sebagai administrator untuk menjalankan perintah ini.

1. Hentikan layanan .

```
sc stop "greengrass"
```

2. Nonaktifkan layanan.

```
sc config "greengrass" start=disabled
```

3. Hapus layanan.

```
sc delete "greengrass"
```

4. Mulai ulang perangkat.

Windows (PowerShell)

> **Note**
>
> Anda harus menjalankan PowerShell sebagai administrator untuk menjalankan perintah ini.

1. Hentikan layanan .

```
Stop-Service -Name "greengrass"
```

2. Nonaktifkan layanan.

```
Set-Service -Name "greengrass" -Status stopped -StartupType disabled
```

3. Hapus layanan.

   - Untuk PowerShell 6.0 dan yang lebih baru:

```
Remove-Service -Name "greengrass" -Confirm:$false -Verbose
```

   - Untuk PowerShell versi yang lebih awal dari 6.0:

```
Get-Item HKLM:\SYSTEM\CurrentControlSet\Services\greengrass | Remove-Item
 -Force -Verbose
```

4. Mulai ulang perangkat.

2. Keluarkan folder root dari perangkat. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder root.

   Linux

```
sudo rm -rf /greengrass/v2
```

   Windows (Command Prompt)

```
rmdir /s /q C:\greengrass\v2
```

   Windows (PowerShell)

```
cmd.exe /c "rmdir /s /q C:\greengrass\v2"
```

3. Hapus perangkat inti dari layanan AWS IoT Greengrass. Langkah ini menghapus informasi status perangkat inti dari AWS Cloud. Pastikan untuk menyelesaikan langkah ini jika Anda berencana untuk menginstal ulang perangkat lunak inti AWS IoT Greengrass ke perangkat inti dengan nama yang sama.

   - Untuk menghapus perangkat inti dari konsol AWS IoT Greengrass tersebut, lakukan hal berikut:

a. Navigasikan ke konsol AWS IoT Greengrass tersebut.
b. Pilih Perangkat inti.
c. Pilih perangkat inti yang akan dikelola.
d. Pilih Hapus.
e. Di modal konfirmasi, pilih Hapus.

- Untuk menghapus perangkat inti denganAWS Command Line Interface, gunakan DeleteCoreDeviceoperasi. Jalankan perintah berikut, dan ganti *MyGreengrassCore*dengan nama perangkat inti.

```
aws greengrassv2 delete-core-device --core-device-thing-name MyGreengrassCore
```

Tutorial AWS IoT Greengrass V2

Anda dapat menyelesaikan tutorial berikut untuk dipelajari AWS IoT Greengrass V2 dan fitur-fiturnya.

Topik

Tutorial: Mengembangkan komponen Greengrass yang menunda pembaruan komponen

Anda dapat menyelesaikan tutorial ini untuk mengembangkan komponen yang menunda pembaruan over-the-air penerapan. Saat menerapkan pembaruan ke perangkat, Anda mungkin ingin menunda pembaruan berdasarkan kondisi, seperti berikut ini:

- Perangkat ini memiliki tingkat baterai rendah.
- Perangkat menjalankan proses atau pekerjaan yang tidak dapat diganggu.
- Perangkat ini memiliki koneksi internet terbatas atau mahal.

> **Note**
>
> Komponen adalah modul perangkat lunak yang berjalan pada perangkat AWS IoT Greengrass inti. Komponen memungkinkan Anda untuk membuat dan mengelola aplikasi yang kompleks sebagai blok bangunan diskrit yang dapat Anda gunakan kembali dari satu perangkat inti Greengrass ke yang lain.

Dalam tutorial ini, Anda melakukan hal-hal berikut:

1. Instal Greengrass Development Kit CLI (GDK CLI) di komputer pengembangan Anda. CLI GDK menyediakan fitur yang membantu Anda mengembangkan komponen Greengrass khusus.
2. Kembangkan komponen Hello World yang menunda pembaruan komponen saat tingkat baterai perangkat inti di bawah ambang batas. Komponen ini berlangganan untuk memperbarui notifikasi menggunakan operasi SubscribeToComponentUpdatesIPC. Saat menerima notifikasi, ia memeriksa apakah level baterai lebih rendah dari ambang batas yang dapat disesuaikan. Jika level baterai di bawah ambang batas, itu menunda pembaruan selama 30 detik menggunakan operasi DeferComponentUpdateIPC. Anda mengembangkan komponen ini di komputer pengembangan Anda menggunakan CLI GDK.

   > **Note**
   >
   > Komponen ini membaca tingkat baterai dari file yang Anda buat pada perangkat inti untuk meniru baterai nyata, sehingga Anda dapat menyelesaikan tutorial ini pada perangkat inti tanpa baterai.

3. Publikasikan komponen itu ke AWS IoT Greengrass layanan.
4. Terapkan komponen itu dari perangkat inti Greengrass AWS Cloud ke untuk mengujinya. Kemudian, Anda memodifikasi level baterai virtual pada perangkat inti, dan membuat penerapan tambahan untuk melihat bagaimana perangkat inti menunda pembaruan saat level baterai rendah.

Anda dapat mengharapkan untuk menghabiskan 20-30 menit pada tutorial ini.

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut:

- Sebuah Akun AWS. Jika Anda tidak memilikinya, lihat Mengatur sebuah Akun AWS.
- Pengguna AWS Identity and Access Management (IAM) dengan izin administrator
- Perangkat inti Greengrass dengan koneksi internet. Untuk informasi lebih lanjut tentang cara menyiapkan perangkat inti, lihat Menyiapkan perangkat AWS IoT Greengrass inti.
  - Python 3.6 atau yang lebih baru diinstal untuk semua pengguna pada perangkat inti dan ditambahkan ke variabel lingkungan. `PATH` Di Windows, Anda juga harus menginstal Python Launcher untuk Windows untuk semua pengguna.

> **Important**
>
> Di Windows, Python tidak menginstal untuk semua pengguna secara default. Ketika Anda menginstal Python, Anda harus menyesuaikan instalasi untuk mengkonfigurasinya untuk perangkat lunak AWS IoT Greengrass Core untuk menjalankan skrip Python. Misalnya, jika Anda menggunakan penginstal Python grafis, lakukan hal berikut:
>
> 1. Pilih Instal peluncur untuk semua pengguna (disarankan).
> 2. Pilih Customize installation.
> 3. Pilih Next.
> 4. Pilih Install for all users.
> 5. Pilih Add Python to environment variables.
> 6. Pilih Instal.
>
> Untuk informasi selengkapnya, lihat Menggunakan Python di Windows dalam dokumentasi Python 3.

- Komputer pengembangan seperti Windows, macOS, atau Unix dengan koneksi internet.
  - Python 3.6 atau yang lebih baru diinstal pada komputer pengembangan Anda.
  - Git diinstal pada komputer pengembangan Anda.
  - AWS Command Line Interface(AWS CLI) diinstal dan dikonfigurasi dengan kredensi pada komputer pengembangan Anda. Untuk informasi selengkapnya, lihat Menginstal, memperbarui, dan melepas pemasangan AWS CLI dan Mengonfigurasi AWS CLI di Panduan Pengguna AWS Command Line Interface.

> **Note**
>
> Jika Anda menggunakan Raspberry Pi atau perangkat ARM 32-bit lainnya, instal V1 AWS CLI. AWS CLI V2 tidak tersedia untuk perangkat ARM 32-bit. Untuk informasi selengkapnya, lihat Menginstal, memperbarui, dan mencopot instalasi AWS CLI versi 1.

Langkah 1: Instal CLI Kit Pengembangan Greengrass

Kit Pengembangan Greengrass CLI (GDK CLI) menyediakan fitur yang membantu Anda mengembangkan komponen Greengrass khusus. Anda dapat menggunakan CLI GDK untuk membuat, membangun, dan menerbitkan komponen kustom.

Jika Anda belum menginstal CLI GDK di komputer pengembangan Anda, selesaikan langkah-langkah berikut untuk menginstalnya.

Untuk menginstal versi terbaru dari CLI GDK

1. Di komputer pengembangan Anda, jalankan perintah berikut untuk menginstal versi terbaru CLI GDK dari GitHub repositorinya.

   ```
   python3 -m pip install -U git+https://github.com/aws-greengrass/aws-greengrass-gdk-cli.git@v1.6.2
   ```

2. Jalankan perintah berikut untuk memverifikasi bahwa CLI GDK berhasil diinstal.

   ```
   gdk --help
   ```

   Jika `gdk` perintah tidak ditemukan, tambahkan foldernya ke PATH.

   - Di perangkat Linux, tambahkan `/home/MyUser/.local/bin` ke PATH, dan ganti *`MyUser`*dengan nama pengguna Anda.
   - Pada perangkat Windows, tambahkan *`PythonPath`*`\\Scripts` ke PATH, dan ganti *`PythonPath`*dengan jalur ke folder Python di perangkat Anda.

Langkah 2: Kembangkan komponen yang menunda pembaruan

Di bagian ini, Anda mengembangkan komponen Hello World dengan Python yang menunda pembaruan komponen saat tingkat baterai perangkat inti berada di bawah ambang batas yang Anda konfigurasikan saat Anda menerapkan komponen. Dalam komponen ini, Anda menggunakan antarmuka interprocess communication (IPC) di AWS IoT Device SDK v2 untuk Python. Anda menggunakan operasi SubscribeToComponentUpdatesIPC untuk menerima pemberitahuan saat perangkat inti menerima penerapan. Kemudian, Anda menggunakan operasi DeferComponentUpdateIPC untuk menunda atau mengakui pembaruan berdasarkan tingkat baterai perangkat.

Untuk mengembangkan komponen Hello World yang menunda pembaruan

1. Di komputer pengembangan Anda, buat folder untuk kode sumber komponen.

```
mkdir com.example.BatteryAwareHelloWorld
cd com.example.BatteryAwareHelloWorld
```

2. Gunakan editor teks untuk membuat file bernamagdk-config.json. CLI GDK membaca dari file konfigurasi CLI GDK, gdk-config.json bernama, untuk membangun dan menerbitkan komponen. File konfigurasi ini ada di root folder komponen.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano gdk-config.json
```

   Salin JSON berikut ke dalam file.

   - Ganti *Amazon* dengan nama Anda.
   - Ganti *us-west-2* dengan Wilayah AWS tempat perangkat inti Anda beroperasi. CLI GDK menerbitkan komponen dalam hal ini. Wilayah AWS
   - Ganti *greengrass-component-artifacts*dengan awalan bucket S3 untuk digunakan. Saat Anda menggunakan CLI GDK untuk mempublikasikan komponen, CLI GDK mengunggah artefak komponen ke bucket S3 yang namanya dibentuk dari nilai ini, file, dan ID Anda menggunakan format Wilayah AWS berikut:. Akun AWS *bucketPrefix*-*region*-*accountId*

     Misalnya, jika Anda menentukan **greengrass-component-artifacts** dan**us-west-2**, dan Akun AWS ID Anda**123456789012**, CLI GDK menggunakan bucket S3 bernama. greengrass-component-artifacts-us-west-2-123456789012

```
{
  "component": {
    "com.example.BatteryAwareHelloWorld": {
      "author": "Amazon",
      "version": "NEXT_PATCH",
      "build": {
        "build_system" : "zip"
      },
```

```
      "publish": {
        "region": "us-west-2",
        "bucket": "greengrass-component-artifacts"
      }
    }
  },
  "gdk_version": "1.0.0"
}
```

File konfigurasi menentukan hal berikut:

- Versi yang akan digunakan saat CLI GDK menerbitkan komponen Greengrass ke layanan cloud. AWS IoT Greengrass `NEXT_PATCH`menentukan untuk memilih versi patch berikutnya setelah versi terbaru yang tersedia di layanan AWS IoT Greengrass cloud. Jika komponen belum memiliki versi di layanan AWS IoT Greengrass cloud, `1.0.0` CLI GDK akan menggunakannya.
- Sistem build untuk komponen. Saat Anda menggunakan sistem `zip` build, CLI GDK mengemas sumber komponen ke dalam file ZIP yang menjadi artefak tunggal komponen.
- Wilayah AWSTempat CLI GDK menerbitkan komponen Greengrass.
- Awalan untuk bucket S3 tempat CLI GDK mengunggah artefak komponen.

3. Gunakan editor teks untuk membuat kode sumber komponen dalam file bernama`main.py`.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   ```
   nano main.py
   ```

   Salin kode Python berikut ke dalam file.

   ```
   import json
   import os
   import sys
   import time
   import traceback

   from pathlib import Path

   from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
   ```

```
HELLO_WORLD_PRINT_INTERVAL = 15  # Seconds
DEFER_COMPONENT_UPDATE_INTERVAL = 30 * 1000  # Milliseconds


class BatteryAwareHelloWorldPrinter():
    def __init__(self, ipc_client: GreengrassCoreIPCClientV2, battery_file_path:
 Path, battery_threshold: float):
        self.battery_file_path = battery_file_path
        self.battery_threshold = battery_threshold
        self.ipc_client = ipc_client
        self.subscription_operation = None

    def on_component_update_event(self, event):
        try:
            if event.pre_update_event is not None:
                if self.is_battery_below_threshold():
                    self.defer_update(event.pre_update_event.deployment_id)
                    print('Deferred update for deployment %s' %
                          event.pre_update_event.deployment_id)
                else:
                    self.acknowledge_update(
                        event.pre_update_event.deployment_id)
                    print('Acknowledged update for deployment %s' %
                          event.pre_update_event.deployment_id)
            elif event.post_update_event is not None:
                print('Applied update for deployment')
        except:
            traceback.print_exc()

    def subscribe_to_component_updates(self):
        if self.subscription_operation == None:
            # SubscribeToComponentUpdates returns a tuple with the response and the
 operation.
            _, self.subscription_operation =
 self.ipc_client.subscribe_to_component_updates(
                on_stream_event=self.on_component_update_event)

    def close_subscription(self):
        if self.subscription_operation is not None:
            self.subscription_operation.close()
            self.subscription_operation = None

    def defer_update(self, deployment_id):
        self.ipc_client.defer_component_update(
```

```
            deployment_id=deployment_id,
 recheck_after_ms=DEFER_COMPONENT_UPDATE_INTERVAL)

    def acknowledge_update(self, deployment_id):
        # Specify recheck_after_ms=0 to acknowledge a component update.
        self.ipc_client.defer_component_update(
            deployment_id=deployment_id, recheck_after_ms=0)

    def is_battery_below_threshold(self):
        return self.get_battery_level() < self.battery_threshold

    def get_battery_level(self):
        # Read the battery level from the virtual battery level file.
        with self.battery_file_path.open('r') as f:
            data = json.load(f)
            return float(data['battery_level'])

    def print_message(self):
        message = 'Hello, World!'
        if self.is_battery_below_threshold():
            message += ' Battery level (%d) is below threshold (%d), so the
 component will defer updates' % (
                self.get_battery_level(), self.battery_threshold)
        else:
            message += ' Battery level (%d) is above threshold (%d), so the
 component will acknowledge updates' % (
                self.get_battery_level(), self.battery_threshold)
        print(message)


def main():
    # Read the battery threshold and virtual battery file path from command-line
 args.
    args = sys.argv[1:]
    battery_threshold = float(args[0])
    battery_file_path = Path(args[1])
    print('Reading battery level from %s and deferring updates when below %d' % (
        str(battery_file_path), battery_threshold))

    try:
        # Create an IPC client and a Hello World printer that defers component
 updates.
        ipc_client = GreengrassCoreIPCClientV2()
        hello_world_printer = BatteryAwareHelloWorldPrinter(
```

```
            ipc_client, battery_file_path, battery_threshold)
        hello_world_printer.subscribe_to_component_updates()
        try:
            # Keep the main thread alive, or the process will exit.
            while True:
                hello_world_printer.print_message()
                time.sleep(HELLO_WORLD_PRINT_INTERVAL)
        except InterruptedError:
            print('Subscription interrupted')
        hello_world_printer.close_subscription()
    except Exception:
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)


if __name__ == '__main__':
    main()
```

Aplikasi Python ini melakukan hal berikut:

- Membaca tingkat baterai perangkat inti dari file tingkat baterai virtual yang akan Anda buat di perangkat inti nanti. File tingkat baterai virtual ini meniru baterai sungguhan, sehingga Anda dapat menyelesaikan tutorial ini pada perangkat inti yang tidak memiliki baterai.
- Membaca argumen baris perintah untuk ambang baterai dan jalur ke file tingkat baterai virtual. Resep komponen menetapkan argumen baris perintah ini berdasarkan parameter konfigurasi, sehingga Anda dapat menyesuaikan nilai-nilai ini saat menerapkan komponen.
- Menggunakan klien IPC V2 di AWS IoT Device SDKv2 untuk Python untuk berkomunikasi dengan perangkat lunak CoreAWS IoT Greengrass. Dibandingkan dengan klien IPC asli, klien IPC V2 mengurangi jumlah kode yang perlu Anda tulis untuk menggunakan IPC dalam komponen khusus.
- Berlangganan untuk memperbarui pemberitahuan menggunakan operasi SubscribeToComponentUpdatesIPC. Perangkat lunak AWS IoT Greengrass Core mengirimkan pemberitahuan sebelum dan sesudah setiap penerapan. Komponen memanggil fungsi berikut setiap kali menerima pemberitahuan. Jika notifikasi untuk penerapan yang akan datang, komponen akan memeriksa apakah level baterai lebih rendah dari ambang batas. Jika level baterai di bawah ambang batas, komponen menunda pembaruan selama 30 detik menggunakan operasi DeferComponentUpdateIPC. Jika tidak, jika tingkat baterai

tidak di bawah ambang batas, komponen mengakui pembaruan, sehingga pembaruan dapat dilanjutkan.

```
def on_component_update_event(self, event):
    try:
        if event.pre_update_event is not None:
            if self.is_battery_below_threshold():
                self.defer_update(event.pre_update_event.deployment_id)
                print('Deferred update for deployment %s' %
                      event.pre_update_event.deployment_id)
            else:
                self.acknowledge_update(
                    event.pre_update_event.deployment_id)
                print('Acknowledged update for deployment %s' %
                      event.pre_update_event.deployment_id)
        elif event.post_update_event is not None:
            print('Applied update for deployment')
    except:
        traceback.print_exc()
```

> **Note**
>
> Perangkat lunak AWS IoT Greengrass Core tidak mengirim pemberitahuan pembaruan untuk penerapan lokal, jadi Anda menerapkan komponen ini menggunakan layanan AWS IoT Greengrass cloud untuk mengujinya.

4. Gunakan editor teks untuk membuat resep komponen dalam file bernama `recipe.json` atau`recipe.yaml`. Resep komponen mendefinisikan metadata komponen, parameter konfigurasi default, dan skrip siklus hidup khusus platform.

   JSON

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   ```
   nano recipe.json
   ```

   Salin JSON berikut ke dalam file.

   ```
   {
   ```

```
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "COMPONENT_NAME",
  "ComponentVersion": "COMPONENT_VERSION",
  "ComponentDescription": "This Hello World component defers updates when the
 battery level is below a threshold.",
  "ComponentPublisher": "COMPONENT_AUTHOR",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "BatteryThreshold": 50,
      "LinuxBatteryFilePath": "/home/ggc_user/virtual_battery.json",
      "WindowsBatteryFilePath": "C:\\Users\\ggc_user\\virtual_battery.json"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk --upgrade",
        "run": "python3 -u {artifacts:decompressedPath}/
com.example.BatteryAwareHelloWorld/main.py \"{configuration:/BatteryThreshold}\"
 \"{configuration:/LinuxBatteryFilePath}\""
      },
      "Artifacts": [
        {
          "Uri": "s3://BUCKET_NAME/COMPONENT_NAME/COMPONENT_VERSION/
com.example.BatteryAwareHelloWorld.zip",
          "Unarchive": "ZIP"
        }
      ]
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk --upgrade",
        "run": "py -3 -u {artifacts:decompressedPath}/
com.example.BatteryAwareHelloWorld/main.py \"{configuration:/BatteryThreshold}\"
 \"{configuration:/WindowsBatteryFilePath}\""
      },
      "Artifacts": [
        {
```

```
          "Uri": "s3://BUCKET_NAME/COMPONENT_NAME/COMPONENT_VERSION/
com.example.BatteryAwareHelloWorld.zip",
          "Unarchive": "ZIP"
        }
      ]
    }
  ]
}
```

YAML

Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano recipe.yaml
```

Salin YAML berikut ke dalam file.

```
---
RecipeFormatVersion: "2020-01-25"
ComponentName: "COMPONENT_NAME"
ComponentVersion: "COMPONENT_VERSION"
ComponentDescription: "This Hello World component defers updates when the
 battery level is below a threshold."
ComponentPublisher: "COMPONENT_AUTHOR"
ComponentConfiguration:
  DefaultConfiguration:
    BatteryThreshold: 50
    LinuxBatteryFilePath: "/home/ggc_user/virtual_battery.json"
    WindowsBatteryFilePath: "C:\\Users\\ggc_user\\virtual_battery.json"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: python3 -m pip install --user awsiotsdk --upgrade
      run: python3 -u {artifacts:decompressedPath}/
com.example.BatteryAwareHelloWorld/main.py "{configuration:/BatteryThreshold}"
 "{configuration:/LinuxBatteryFilePath}"
    Artifacts:
      - Uri: "s3://BUCKET_NAME/COMPONENT_NAME/COMPONENT_VERSION/
com.example.BatteryAwareHelloWorld.zip"
        Unarchive: ZIP
```

```
  - Platform:
      os: windows
    Lifecycle:
      install: py -3 -m pip install --user awsiotsdk --upgrade
      run: py -3 -u {artifacts:decompressedPath}/
com.example.BatteryAwareHelloWorld/main.py "{configuration:/BatteryThreshold}"
 "{configuration:/WindowsBatteryFilePath}"
    Artifacts:
      - Uri: "s3://BUCKET_NAME/COMPONENT_NAME/COMPONENT_VERSION/
com.example.BatteryAwareHelloWorld.zip"
        Unarchive: ZIP
```

Resep ini menentukan yang berikut:

- Parameter konfigurasi default untuk ambang baterai, jalur file baterai virtual pada perangkat inti Linux, dan jalur file baterai virtual pada perangkat inti Windows.
- `install`Siklus hidup yang menginstal versi terbaru v2 AWS IoT Device SDK untuk Python.
- `run`Siklus hidup yang menjalankan aplikasi Python di. `main.py`
- Placeholder, seperti `COMPONENT_NAME` dan`COMPONENT_VERSION`, di mana CLI GDK menggantikan informasi saat membangun resep komponen.

Untuk informasi lebih lanjut tentang resep komponen, lihatAWS IoT Greengrass referensi resep komponen.

Langkah 3: Publikasikan komponen ke AWS IoT Greengrass layanan

Di bagian ini, Anda mempublikasikan komponen Hello World ke layanan AWS IoT Greengrass cloud. Setelah komponen tersedia di layanan AWS IoT Greengrass cloud, Anda dapat menerapkannya ke perangkat inti. Anda menggunakan CLI GDK untuk mempublikasikan komponen dari komputer pengembangan Anda ke layanan cloudAWS IoT Greengrass. CLI GDK mengunggah resep dan artefak komponen untuk Anda.

Untuk mempublikasikan komponen Hello World ke AWS IoT Greengrass layanan

1. Jalankan perintah berikut untuk membangun komponen menggunakan CLI GDK. Perintah pembuatan komponen membuat resep dan artefak berdasarkan file konfigurasi CLI GDK. Dalam proses ini, CLI GDK membuat file ZIP yang berisi kode sumber komponen.

```
gdk component build
```

Anda akan melihat pesan yang mirip dengan contoh berikut ini.

```
[2022-04-28 11:20:16] INFO - Getting project configuration from gdk-config.json
[2022-04-28 11:20:16] INFO - Found component recipe file 'recipe.yaml' in the
 project directory.
[2022-04-28 11:20:16] INFO - Building the component
 'com.example.BatteryAwareHelloWorld' with the given project configuration.
[2022-04-28 11:20:16] INFO - Using 'zip' build system to build the component.
[2022-04-28 11:20:16] WARNING - This component is identified as using 'zip' build
 system. If this is incorrect, please exit and specify custom build command in the
 'gdk-config.json'.
[2022-04-28 11:20:16] INFO - Zipping source code files of the component.
[2022-04-28 11:20:16] INFO - Copying over the build artifacts to the greengrass
 component artifacts build folder.
[2022-04-28 11:20:16] INFO - Updating artifact URIs in the recipe.
[2022-04-28 11:20:16] INFO - Creating component recipe in 'C:\Users\finthomp
\greengrassv2\com.example.BatteryAwareHelloWorld\greengrass-build\recipes'.
```

2. Jalankan perintah berikut untuk mempublikasikan komponen ke layanan AWS IoT Greengrass cloud. Perintah component publish mengunggah artefak file ZIP komponen ke bucket S3. Kemudian, ia memperbarui URI S3 file ZIP dalam resep komponen dan mengunggah resep ke layanan. AWS IoT Greengrass Dalam proses ini, CLI GDK memeriksa versi komponen Hello World apa yang sudah tersedia di AWS IoT Greengrass layanan cloud, sehingga dapat memilih versi patch berikutnya setelah versi itu. Jika komponen belum ada, CLI GDK menggunakan versi. `1.0.0`

   ```
   gdk component publish
   ```

   Anda akan melihat pesan yang mirip dengan contoh berikut ini. Output memberi tahu Anda versi komponen yang dibuat CLI GDK.

   ```
   [2022-04-28 11:20:29] INFO - Getting project configuration from gdk-config.json
   [2022-04-28 11:20:29] INFO - Found component recipe file 'recipe.yaml' in the
    project directory.
   [2022-04-28 11:20:29] INFO - Found credentials in shared credentials file: ~/.aws/
   credentials
   ```

```
[2022-04-28 11:20:30] INFO - No private version of the component
 'com.example.BatteryAwareHelloWorld' exist in the account. Using '1.0.0' as the
 next version to create.
[2022-04-28 11:20:30] INFO - Publishing the component
 'com.example.BatteryAwareHelloWorld' with the given project configuration.
[2022-04-28 11:20:30] INFO - Uploading the component built artifacts to s3 bucket.
[2022-04-28 11:20:30] INFO - Uploading component artifacts to S3
 bucket: greengrass-component-artifacts-us-west-2-123456789012. If this is your
 first time using this bucket, add the 's3:GetObject' permission to each core
 device's token exchange role to allow it to download the component artifacts. For
 more information, see https://docs.aws.amazon.com/greengrass/v2/developerguide/
device-service-role.html.
[2022-04-28 11:20:30] INFO - Not creating an artifacts bucket as it already exists.
[2022-04-28 11:20:30] INFO - Updating the component recipe
 com.example.BatteryAwareHelloWorld-1.0.0.
[2022-04-28 11:20:31] INFO - Creating a new greengrass component
 com.example.BatteryAwareHelloWorld-1.0.0
[2022-04-28 11:20:31] INFO - Created private version '1.0.0' of the component in
 the account.'com.example.BatteryAwareHelloWorld'.
```

3. Salin nama bucket S3 dari output. Anda menggunakan nama bucket nanti untuk memungkinkan perangkat inti mengunduh artefak komponen dari bucket ini.
4. (Opsional) Lihat komponen di AWS IoT Greengrass konsol untuk memverifikasi bahwa komponen tersebut berhasil diunggah. Lakukan hal-hal berikut:

   a. Pada menu navigasi konsol AWS IoT Greengrass tersebut, pilih Komponen.

   b. Pada halaman Components, pilih tab My components, lalu pilih com.example.BatteryAwareHelloWorld.

      Pada halaman ini, Anda dapat melihat resep komponen dan informasi lain tentang komponen.
5. Izinkan perangkat inti mengakses artefak komponen dalam bucket S3.

   Setiap perangkat inti memiliki peran IAM perangkat inti yang memungkinkannya berinteraksi dengan AWS IoT dan mengirim log ke AWS Cloud. Peran perangkat ini tidak mengizinkan akses ke bucket S3 secara default, sehingga Anda harus membuat dan melampirkan kebijakan yang memungkinkan perangkat inti mengambil artefak komponen dari bucket S3.

   Jika peran perangkat Anda sudah memungkinkan akses ke bucket S3, Anda dapat melewati langkah ini. Jika tidak, buat kebijakan IAM yang memungkinkan akses dan melampirkannya pada peran, sebagai berikut:

a. Di menu navigasi konsol IAM, pilih Kebijakan, lalu pilih Buat kebijakan.

b. Pada tab JSON, ganti placeholder konten dengan kebijakan berikut. Ganti *greengrass-component-artifacts-us-west-2-123456789012* dengan nama bucket S3 tempat CLI GDK mengunggah artefak komponen.

   Misalnya, jika Anda menentukan **greengrass-component-artifacts** dan **us-west-2** dalam file konfigurasi CLI GDK, dan ID Akun AWS **123456789012** Anda, CLI GDK menggunakan bucket S3 bernama. greengrass-component-artifacts-us-west-2-123456789012

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::greengrass-component-artifacts-us-
west-2-123456789012/*"
    }
  ]
}
```

c. Pilih Berikutnya.

d. Di bagian Detail kebijakan, untuk Nama, masukkan**MyGreengrassV2ComponentArtifactPolicy**.

e. Pilih Buat kebijakan.

f. Di menu navigasi konsol IAM, pilih Peran, lalu pilih nama peran untuk perangkat inti. Anda menentukan nama peran ini saat menginstal perangkat lunak AWS IoT Greengrass Inti. Jika Anda tidak menentukan nama, defaultnya adalahGreengrassV2TokenExchangeRole.

g. Di bawah Izin, pilih Tambahkan izin, lalu pilih Lampirkan kebijakan.

h. Pada halaman Tambahkan izin, pilih kotak centang di samping MyGreengrassV2ComponentArtifactPolicy kebijakan yang Anda buat, lalu pilih Tambahkan izin.

Langkah 4: Menyebarkan dan menguji komponen pada perangkat inti

Di bagian ini, Anda menyebarkan komponen ke perangkat inti untuk menguji fungsinya. Pada perangkat inti, Anda membuat file tingkat baterai virtual untuk meniru baterai nyata. Kemudian, Anda membuat penerapan tambahan dan mengamati file log komponen pada perangkat inti untuk melihat komponen menunda dan mengakui pembaruan.

Untuk menyebarkan dan menguji komponen Hello World yang menunda pembaruan

1. Gunakan editor teks untuk membuat file tingkat baterai virtual. File ini meniru baterai sungguhan.

   - Pada perangkat inti Linux, buat file bernama`/home/ggc_user/virtual_battery.json`. Jalankan editor teks dengan `sudo` izin.
   - Pada perangkat inti Windows, buat file bernama`C:\Users\ggc_user\virtual_battery.json`. Jalankan editor teks sebagai administrator.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   ```
   sudo nano /home/ggc_user/virtual_battery.json
   ```

   Salin JSON berikut ke dalam file.

   ```
   {
     "battery_level": 50
   }
   ```

2. Menyebarkan komponen Hello World ke perangkat inti. Lakukan hal-hal berikut:

   a. Pada menu navigasi konsol AWS IoT Greengrass tersebut, pilih Komponen.

   b. Pada halaman Components, pilih tab My components, lalu pilih com.example.BatteryAwareHelloWorld.

   c. Pada halaman com.example.BatteryAwareHelloWorld pilih Deploy.

   d. Dari Tambahkan ke penerapan, pilih penerapan yang ada untuk direvisi, atau pilih untuk membuat penerapan baru, lalu pilih Berikutnya.

e. Jika Anda memilih untuk membuat penerapan baru, pilih perangkat inti target atau grup hal untuk penerapan. Pada halaman Tentukan target, di bawah target Deployment, pilih perangkat inti atau grup benda, lalu pilih Berikutnya.

f. Pada halaman Pilih komponen, verifikasi bahwa com.example.BatteryAwareHelloWorldkomponen dipilih, pilih Berikutnya.

g. Pada halaman Configure components, pilih com.example.BatteryAwareHelloWorld, lalu lakukan hal berikut:

   i. Pilih Konfigurasi komponen.

   ii. Dalam konfigurasi com.example.BatteryAwareHelloWorld modal, di bawah Configuration update, di Configuration to merge, masukkan update konfigurasi berikut.

```
{
  "BatteryThreshold": 70
}
```

   iii. Pilih Konfirmasi untuk menutup modal, lalu pilih Berikutnya.

h. Pada halaman Konfirmasi pengaturan lanjutan, di bagian Kebijakan penerapan, di bawah Kebijakan pembaruan komponen, konfirmasikan bahwa komponen Beri tahu dipilih. Notify komponen dipilih secara default saat Anda membuat penerapan baru.

i. Di halaman Tinjau, pilih Deploy.

   Penyebaran dapat memakan waktu hingga satu menit untuk diselesaikan.

3. Perangkat lunak AWS IoT Greengrass Core menyimpan stdout dari proses komponen ke file log di folder. `logs` Jalankan perintah berikut untuk memverifikasi bahwa komponen Hello World berjalan dan mencetak pesan status.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

PowerShell

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

Anda akan melihat pesan yang mirip dengan contoh berikut ini.

```
Hello, World! Battery level (50) is below threshold (70), so the component will
 defer updates.
```

> **Note**
>
> Jika file tidak ada, penerapan mungkin belum lengkap. Jika file tidak ada dalam waktu 30 detik, penerapan kemungkinan gagal. Ini dapat terjadi jika perangkat inti tidak memiliki izin untuk mengunduh artefak komponen dari bucket S3, misalnya. Jalankan perintah berikut untuk melihat file log perangkat lunak AWS IoT Greengrass inti. File ini mencakup log dari layanan deployment perangkat inti Greengrass.
>
> Linux or Unix
>
> ```
> sudo tail -f /greengrass/v2/logs/greengrass.log
> ```
>
> Windows Command Prompt (CMD)
>
> ```
> type C:\greengrass\v2\logs\greengrass.log
> ```
>
> `type`Perintah menulis konten file ke terminal. Jalankan perintah ini beberapa kali untuk mengamati perubahan dalam file.
>
> PowerShell
>
> ```
> gc C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
> ```

4. Buat penerapan baru ke perangkat inti untuk memverifikasi bahwa komponen menunda pembaruan. Lakukan hal-hal berikut:

   a. Di menu navigasi AWS IoT Greengrasskonsol, pilih Deployment.

   b. Pilih penerapan yang Anda buat atau revisi sebelumnya.

c. Pada halaman penyebaran, pilih Revisi.

d. Dalam modal Revise deployment, pilih Revise deployment.

e. Pilih Berikutnya di setiap langkah, lalu pilih Deploy.

5. Jalankan perintah berikut untuk melihat log komponen lagi, dan verifikasi bahwa itu menunda pembaruan.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

   Anda akan melihat pesan yang mirip dengan contoh berikut ini. Komponen menunda pembaruan selama 30 detik, sehingga komponen mencetak pesan ini berulang kali.

   ```
   Deferred update for deployment 50722a95-a05f-4e2a-9414-da80103269aa.
   ```

6. Gunakan editor teks untuk mengedit file tingkat baterai virtual dan mengubah tingkat baterai ke nilai di atas ambang batas, sehingga penerapan dapat dilanjutkan.

   - Pada perangkat inti Linux, edit file bernama`/home/ggc_user/virtual_battery.json`. Jalankan editor teks dengan `sudo` izin.
   - Pada perangkat inti Windows, edit file bernama`C:\Users\ggc_user\virtual_battery.json`. Jalankan editor teks sebagai administrator.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   ```
   sudo nano /home/ggc_user/virtual_battery.json
   ```

Ubah level baterai menjadi80.

```
{
  "battery_level": 80
}
```

7. Jalankan perintah berikut untuk melihat log komponen lagi, dan verifikasi bahwa ia mengakui pembaruan.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

   Anda akan melihat pesan yang mirip dengan contoh berikut.

   ```
   Hello, World! Battery level (80) is above threshold (70), so the component will
    acknowledge updates.
   Acknowledged update for deployment f9499eb2-4a40-40a7-86c1-c89887d859f1.
   ```

Anda telah menyelesaikan tutorial ini. Komponen Hello World menunda atau mengakui pembaruan berdasarkan tingkat baterai perangkat inti. Untuk informasi selengkapnya tentang topik yang dieksplorasi pada tutorial ini, lihat berikut ini:

- Kembangkan AWS IoT Greengrass komponen
- Deploy komponen AWS IoT Greengrass ke perangkat
- Gunakan AWS IoT Device SDK untuk berkomunikasi dengan inti Greengrass, komponen lain, dan AWS IoT Core
- AWS IoT GreengrassKit Pengembangan Antarmuka Baris Perintah

Tutorial: Berinteraksi dengan perangkat IoT lokal melalui MQTT

Anda dapat menyelesaikan tutorial ini untuk mengonfigurasi perangkat inti untuk berinteraksi dengan perangkat IoT lokal, yang disebut perangkat klien, yang terhubung ke perangkat inti melalui MQTT. Dalam tutorial ini, Anda mengonfigurasi objek AWS IoT untuk menggunakan penemuan cloud untuk terhubung ke perangkat inti sebagai perangkat klien. Bila Anda mengonfigurasi penemuan cloud, perangkat klien dapat mengirim permintaan ke layanan cloud AWS IoT Greengrass untuk menemukan perangkat inti. Tanggapan dari AWS IoT Greengrass mencakup informasi konektivitas dan sertifikat untuk perangkat inti yang Anda konfigurasikan perangkat kliennya untuk ditemukan. Kemudian, perangkat klien dapat menggunakan informasi ini untuk menyambung ke perangkat inti yang tersedia di mana ia dapat berkomunikasi melalui MQTT.

Dalam tutorial ini, Anda melakukan hal-hal berikut:

1. Meninjau dan memperbarui izin perangkat inti, jika diperlukan.
2. Mengaitkan perangkat klien ke perangkat inti, sehingga dapat menemukan perangkat inti menggunakan penemuan cloud.
3. Menyebarkan komponen Greengrass ke perangkat inti untuk mengaktifkan dukungan perangkat klien.
4. Hubungkan perangkat klien ke perangkat inti dan uji komunikasi dengan layanan cloud AWS IoT Core.
5. Kembangkan komponen Greengrass khusus yang berkomunikasi dengan perangkat klien.
6. Kembangkan komponen khusus yang berinteraksi dengan bayangan perangkat perangkat klien. AWS IoT

Tutorial ini menggunakan perangkat inti tunggal dan perangkat klien tunggal. Anda juga dapat mengikuti tutorial untuk menghubungkan dan menguji beberapa perangkat klien.

Anda dapat mengharapkan untuk menghabiskan 30-60 menit pada tutorial ini.

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut:

- Sebuah Akun AWS. Jika Anda tidak memilikinya, lihat Mengatur sebuah Akun AWS.
- Pengguna (IAM).AWS Identity and Access Management dengan izin administrator.

- Sebuah perangkat inti Greengrass. Untuk informasi lebih lanjut tentang cara menyiapkan perangkat inti, lihat Menyiapkan perangkat AWS IoT Greengrass inti.
  - Perangkat inti harus menjalankan Greengrass nucleus v2.6.0 atau yang lebih baru. Versi ini mencakup dukungan untuk wildcard dalam komunikasi penerbitan/berlangganan lokal dan dukungan untuk bayangan perangkat klien.

    > **Note**
    >
    > Dukungan perangkat klien memerlukan Greengrass nucleus v2.2.0 atau yang lebih baru. Namun, tutorial ini mengeksplorasi fitur yang lebih baru, seperti dukungan untuk wildcard MQTT di penerbitan/berlangganan lokal dan dukungan untuk bayangan perangkat klien. Fitur-fitur ini membutuhkan Greengrass nucleus v2.6.0 atau yang lebih baru.

  - Perangkat inti harus berada di jaringan yang sama dengan perangkat klien untuk terhubung.
  - (Opsional) Untuk menyelesaikan modul tempat Anda mengembangkan komponen Greengrass khusus, perangkat inti harus menjalankan CLI Greengrass. Untuk informasi selengkapnya, lihat Instal Greengrass CLI.
- Sesi objek AWS IoT untuk terhubung sebagai perangkat klien dalam tutorial ini. Untuk informasi lebih lanjut, lihat Buat AWS IoT sumber daya di AWS IoT Core Panduan Developer.
  - Kebijakan AWS IoT perangkat klien harus memungkinkan izin `greengrass:Discover`. Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat klien.
  - Perangkat klien harus berada di jaringan yang sama dengan perangkat inti.
  - Perangkat klien harus menjalankan Python 3.
  - Perangkat klien harus menjalankan Git.

Langkah 1: Tinjau dan perbarui AWS IoT kebijakan perangkat inti

Untuk mendukung perangkat klien, kebijakan AWS IoT harus mengizinkan izin berikut:

- `greengrass:PutCertificateAuthorities`
- `greengrass:VerifyClientDeviceIdentity`
- `greengrass:VerifyClientDeviceIoTCertificateAssociation`
- `greengrass:GetConnectivityInfo`

- `greengrass:UpdateConnectivityInfo` – (Opsional) Izin ini diperlukan untuk menggunakan komponen detektor IP, yang melaporkan informasi konektivitas jaringan perangkat inti ke layanan cloud AWS IoT Greengrass.

Untuk informasi selengkapnya tentang izin dan AWS IoT kebijakan ini untuk perangkat inti, lihat Kebijakan AWS IoT untuk operasi bidang data danKebijakan AWS IoT minimal untuk mendukung perangkat klien.

Di bagian ini, Anda meninjau AWS IoT untuk perangkat inti Anda dan menambahkan izin yang diperlukan yang hilang. Jika Anda menggunakan Penginstal perangkat lunak inti AWS IoT Greengrass untuk menyediakan sumber daya, perangkat inti Anda memiliki kebijakan AWS IoT yang mengizinkan akses ke semua tindakan AWS IoT Greengrass (`greengrass:*`). Dalam hal ini, Anda harus memperbarui AWS IoT kebijakan hanya jika Anda berencana untuk mengonfigurasi komponen pengelola bayangan untuk menyinkronkan bayangan perangkatAWS IoT Core. Jika tidak, Anda dapat melewati bagian ini.

Untuk meninjau dan memperbarui kebijakan AWS IoT perangkat inti

1. Di menu navigasi konsol AWS IoT Greengrass tersebut, pilih Perangkat inti.
2. Pada halaman Perangkat inti, pilih perangkat inti yang akan diperbarui.
3. Pada halaman detail perangkat inti, pilih tautan ke Objek perangkat inti. Tautan ini membuka halaman rincian hal di AWS IoT konsol.
4. Pada halaman detail objek, pilih Sertifikat.
5. Di tab Sertifikat, pilih sertifikat aktif objek.
6. Pada halaman detail sertifikat, pilih Kebijakan.
7. Di tab Kebijakan, pilih kebijakan AWS IoT yang akan ditinjau dan diperbarui. Anda dapat menambahkan izin yang diperlukan untuk kebijakan yang dilampirkan ke sertifikat aktif perangkat inti.

> **Note**
>
> Jika Anda menggunakan Penginstal perangkat lunak inti AWS IoT Greengrass untuk menyediakan sumber daya, Anda memiliki dua kebijakan AWS IoT. Kami menyarankan Anda memilih kebijakan yang diberi nama GreengrassV2IoTThingPolicy, jika ada. Perangkat inti yang Anda buat dengan penginstal cepat menggunakan nama kebijakan

ini secara default. Jika Anda menambahkan izin untuk kebijakan ini, Anda juga memberikan izin ini ke perangkat inti lain yang menggunakan kebijakan ini.

8. Dalam ikhtisar kebijakan, pilih Edit versi aktif.
9. Tinjau kebijakan untuk izin yang diperlukan, dan tambahkan izin yang diperlukan yang hilang.
10. Untuk menetapkan versi kebijakan baru sebagai versi aktif, di bawah Status versi Kebijakan, pilih Setel versi yang diedit sebagai versi aktif untuk kebijakan ini.
11. Pilih Simpan sebagai versi baru.

Langkah 2: Aktifkan dukungan perangkat klien

Untuk perangkat klien untuk menggunakan penemuan cloud untuk menyambung ke perangkat inti, Anda harus mengaitkan perangkat itu. Ketika Anda mengaitkan perangkat klien ke perangkat inti, Anda mengaktifkan perangkat klien untuk mengambil alamat IP perangkat inti dan sertifikat untuk digunakan untuk menyambung.

Untuk memungkinkan perangkat klien untuk aman terhubung ke perangkat inti dan berkomunikasi dengan komponen Greengrass dan AWS IoT Core, Anda men-deploy komponen Greengrass berikut ke perangkat inti:

- Auth perangkat klien (`aws.greengrass.clientdevices.Auth`)

  Deploy komponen auth perangkat klien untuk mengautentikasi perangkat klien dan mengotorisasi tindakan perangkat klien. Komponen ini memungkinkan objek AWS IoT Anda untuk terhubung ke perangkat inti.

  Komponen ini memerlukan beberapa konfigurasi untuk menggunakannya. Anda harus menentukan grup perangkat klien dan operasi yang setiap grup diotorisasi untuk melakukan, seperti untuk menghubungkan dan berkomunikasi melalui MQTT. Untuk informasi lebih lanjut, lihat konfigurasi komponen auth perangkat klien.
- MQTT 3.1.1 broker (Moquette) (`aws.greengrass.clientdevices.mqtt.Moquette`)

  Menyebarkan komponen broker Moquette MQTT untuk menjalankan broker MQTT ringan. Broker Moquette MQTT patuh pada MQTT 3.1.1 dan mencakup dukungan lokal untuk QoS 0, QoS 1, QoS 2, pesan yang dipertahankan, pesan kehendak terakhir, dan langganan terus-menerus.

Anda tidak diharuskan mengonfigurasi komponen ini untuk menggunakannya. Namun, Anda dapat mengonfigurasi port di mana komponen ini mengoperasikan broker MQTT. Secara default, ia menggunakan port 8883.

- Jembatan MQTT (`aws.greengrass.clientdevices.mqtt.Bridge`)

  (Opsional) Deploy komponen jembatan MQTT untuk merelai pesan antar perangkat klien (MQTT lokal), penerbitan/berlangganan lokal, dan MQTT AWS IoT Core. Konfigurasikan komponen ini untuk menyinkronkan perangkat klien dengan AWS IoT Core dan berinteraksi dengan perangkat klien dari komponen Greengrass.

  Komponen ini memerlukan konfigurasi untuk digunakan. Anda harus menentukan pemetaan topik di mana komponen ini merelai pesan. Untuk informasi lebih lanjut, lihat konfigurasi komponen jembatan MQTT.

- Detektor IP (`aws.greengrass.clientdevices.IPDetector`)

  (Opsional) Deploy komponen detektor IP untuk secara otomatis melaporkan titik akhir broker MQTT perangkat inti ke layanan cloud AWS IoT Greengrass. Anda tidak dapat menggunakan komponen ini jika Anda memiliki pengaturan jaringan yang kompleks, seperti di mana router meneruskan port broker MQTT ke perangkat inti.

  Anda tidak diharuskan mengonfigurasi komponen ini untuk menggunakannya.

Di bagian ini, Anda menggunakan konsol AWS IoT Greengrass untuk mengaitkan perangkat klien dan men-deploy komponen perangkat klien ke perangkat inti.

Untuk mengaktifkan dukungan perangkat klien

1. Navigasikan ke konsol AWS IoT Greengrass tersebut.
2. Di menu navigasi kiri, pilih Perangkat inti.
3. Pada halaman Perangkat Inti, pilih perangkat inti tempat Anda ingin mengaktifkan dukungan perangkat klien.
4. Pada halaman detail perangkat inti, pilih tab Perangkat klien.
5. Pada tab Perangkat klien, pilih Konfigurasikan penemuan cloud.

   Halaman Konfigurasikan penemuan perangkat inti terbuka. Di bagian ini, Anda mengaitkan perangkat klien ke perangkat inti dan men-deploy komponen perangkat klien. Halaman ini memilih perangkat inti untuk Anda di Langkah 1: Pilih perangkat inti target.

> **Note**
>
> Anda juga dapat menggunakan halaman ini untuk mengonfigurasi penemuan perangkat inti untuk grup objek. Jika Anda memilih opsi ini, Anda dapat men-deploy komponen perangkat klien ke semua perangkat inti dalam grup objek. Namun, jika Anda memilih opsi ini, Anda harus secara manual mengaitkan perangkat klien untuk setiap perangkat inti kemudian setelah Anda membuat deployment. Dalam tutorial ini, Anda mengonfigurasi perangkat inti tunggal.

6. Di Langkah 2: Kaitkan perangkat klien, kaitkan objek AWS IoT ke perangkat inti. Hal ini memungkinkan perangkat klien untuk menggunakan penemuan cloud untuk mengambil informasi konektivitas perangkat inti dan sertifikat. Lakukan hal-hal berikut:

   a. Pilih Kaitkan perangkat klien.

   b. Di modal Kaitkan perangkat klien dengan perangkat inti, masukkan nama objek AWS IoT yang akan dikaitkan.

   c. Pilih Tambahkan.

   d. Pilih Kaitkan.

7. Di Langkah 3: Konfigurasi dan deploy komponen Greengrass, deploy komponen untuk mengaktifkan dukungan perangkat klien. Jika perangkat inti target memiliki deployment sebelumnya, halaman ini merevisi deployment tersebut. Jika tidak, halaman ini membuat deployment baru untuk perangkat inti. Lakukan hal berikut untuk mengonfigurasi dan men-deploy komponen perangkat klien:

   a. Perangkat inti harus menjalankan Greengrass nucleus v2.6.0 atau yang lebih baru untuk menyelesaikan tutorial ini. Jika perangkat inti menjalankan versi sebelumnya, lakukan hal berikut:

      i. Pilih kotak untuk menyebarkan aws.greengrass.Nucleuskomponen.

      ii. Untuk komponen aws.greengrass.Nucleus, pilih Edit konfigurasi.

      iii. Untuk versi Komponen, pilih versi 2.6.0 atau yang lebih baru.

      iv. Pilih Konfirmasi.

> **Note**
>
> Jika Anda memutakhirkan inti Greengrass dari versi minor sebelumnya, dan perangkat inti AWSmenjalankan komponen yang disediakan yang bergantung pada nukleus, Anda juga harus memperbarui AWS komponen yang disediakan ke versi yang lebih baru. Anda dapat mengonfigurasi versi komponen ini ketika Anda meninjau deployment nanti dalam tutorial ini. Untuk informasi selengkapnya, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

b. Untuk komponen aws.greengrass.clientdevices.Auth, pilih Edit konfigurasi.

c. Di modal Edit konfigurasi untuk komponen auth perangkat klien, konfigurasi kebijakan otorisasi yang memungkinkan perangkat klien untuk mempublikasikan dan berlangganan broker MQTT pada perangkat inti. Lakukan hal-hal berikut:

   i. Di bawah Konfigurasi, di blok Konfigurasi untuk menggabungkan kode, masukkan konfigurasi berikut, yang berisi kebijakan otorisasi perangkat klien. Setiap perangkat grup otorisasi kebijakan menentukan serangkaian tindakan dan sumber daya di mana perangkat klien dapat melakukan tindakan tersebut.

   - Kebijakan ini memungkinkan perangkat klien yang namanya dimulai dengan `MyClientDevice` untuk menghubungkan dan berkomunikasi pada semua topik MQTT. *`MyClientDeviceGanti*`* dengan nama AWS IoT benda yang akan dihubungkan sebagai perangkat klien. Anda juga dapat menentukan nama dengan wldcard `*` yang cocok dengan nama perangkat klien. Wildcard `*` harus di akhir nama.

     Jika Anda memiliki perangkat klien kedua untuk disambungkan, *`MyOtherClientDeviceganti*`* dengan nama perangkat klien tersebut, atau pola wildcard yang cocok dengan nama perangkat klien tersebut. Jika tidak, Anda dapat menghapus atau menyimpan bagian ini dari aturan pemilihan yang memungkinkan perangkat klien dengan nama yang cocok dengan `MyOtherClientDevice*` untuk terhubung dan berkomunikasi.

   - Kebijakan ini menggunakan operator `OR` untuk juga memungkinkan perangkat klien yang namanya dimulai dengan `MyOtherClientDevice` untuk menghubungkan dan berkomunikasi pada semua topik MQTT. Anda dapat menghapus klausul ini dalam aturan seleksi atau memodifikasinya agar cocok dengan perangkat klien untuk terhubung.

- Kebijakan ini memungkinkan perangkat klien untuk mempublikasikan dan berlangganan pada semua topik MQTT. Untuk mengikuti praktik keamanan terbaik, batasi operasi `mqtt:publish` dan `mqtt:subscribe` ke set minimal topik yang diagunakan oleh perangkat klien untuk berkomunikasi.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice* OR
 thingName: MyOtherClientDevice*",
        "policyName": "MyClientDevicePolicy"
      }
    },
    "policies": {
      "MyClientDevicePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish to all
 topics.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to all
 topics.",
          "operations": [
            "mqtt:subscribe"
          ],
```

```
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

Untuk informasi lebih lanjut, lihat Konfigurasi komponen auth perangkat klien.

ii. Pilih Konfirmasi.

d. Untuk komponen aws.greengrass.clientdevices.mqtt.Bridge, pilih Edit konfigurasi.

e. Di modal Edit konfigurasi untuk komponen jembatan MQTT, konfigurasi pemetaan topik yang menyampaikan pesan MQTT dari perangkat klien ke AWS IoT Core. Lakukan hal-hal berikut:

i. Di bawah Konfigurasi, di blok kode Konfigurasi yang akan digabungkan, masukkan konfigurasi berikut. Konfigurasi ini menentukan untuk merelai pesan MQTT pada topik filter `clients/+/hello/world` dari perangkat klien ke layanan cloud AWS IoT Core. Sebagai contoh, filter topik ini cocok dengan topik `clients/MyClientDevice1/hello/world`.

```
{
  "mqttTopicMapping": {
    "HelloWorldIotCoreMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

Untuk informasi lebih lanjut, lihat konfigurasi komponen jembatan MQTT.

ii. Pilih Konfirmasi.

8. Pilih Tinjau dan deploy untuk meninjau deployment yang dibuat oleh halaman ini untuk Anda.

9. Jika sebelumnya Anda belum mengatur Peran layanan Greengrass di Wilayah ini, konsol membuka modal untuk mengatur peran layanan untuk Anda. Komponen autentikasi perangkat klien menggunakan peran layanan ini untuk memverifikasi identitas perangkat klien, dan

komponen detektor IP menggunakan peran layanan ini untuk mengelola informasi konektivitas perangkat inti. Pilih Berikan izin.

10. Pada halaman Tinjauan, pilih Deploy untuk memulai deployment ke perangkat inti.
11. Untuk memverifikasi bahwa deployment berhasil, periksa status deployment, dan periksa log pada perangkat inti. Untuk memeriksa status deployment pada perangkat inti, Anda dapat memilih Target pada Gambaran Umum deployment. Untuk informasi selengkapnya, lihat hal berikut:

    - Periksa status deployment
    - Memantau AWS IoT Greengrass log

Langkah 3: Connect perangkat klien

Perangkat klien dapat menggunakan AWS IoT Device SDK untuk menemukan, menghubungkan, dan berkomunikasi dengan perangkat inti. Perangkat klien harus menjadi AWS IoT sesuatu. Untuk informasi selengkapnya, lihat Membuat objek benda di Panduan AWS IoT Core Pengembang.

Di bagian ini, Anda menginstal v2 for Python AWS IoT Device SDK dan menjalankan aplikasi contoh penemuan Greengrass dari AWS IoT Device SDK.

> **Note**
>
> AWS IoT Device SDK juga tersedia dalam bahasa pemrograman lain. Tutorial ini menggunakan v2 for Python AWS IoT Device SDK, tetapi Anda dapat menjelajahi SDK lain untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat AWS IoT SDK Perangkat di AWS IoT Core Panduan Developer.

Untuk menyambungkan perangkat klien ke perangkat inti

1. Unduh dan instal v2 for Python AWS IoT Device SDK ke objek AWS IoT untuk terhubung sebagai perangkat klien.

    Di perangkat klien, lakukan hal berikut:

    a. Kloning v2 for Python repository AWS IoT Device SDK untuk mengunduhnya.

    ```
    git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
    ```

b. Instal v2 for Python AWS IoT Device SDK.

```
python3 -m pip install --user ./aws-iot-device-sdk-python-v2
```

2. Ubah ke folder contoh di v2 for Python AWS IoT Device SDK.

```
cd aws-iot-device-sdk-python-v2/samples
```

3. Jalankan sampel aplikasi penemuan Greengrass. Aplikasi ini mengharapkan argumen yang menentukan nama objek perangkat klien, topik MQTT dan pesan yang akan digunakan, dan sertifikat yang mengautentikasi dan mengamankan sambungan. Contoh berikut mengirimkan pesan Hello World ke topik `clients/`*`MyClientDevice1`*`/hello/world`.

> **Note**
>
> Topik ini cocok dengan topik tempat Anda mengonfigurasi jembatan MQTT untuk menyampaikan pesan sebelumnya. AWS IoT Core

- Ganti *`MyClientDevice1`* dengan nama benda perangkat klien.
- Ganti *`~/certs/ AmazonRoot ca1.pem dengan`* jalur ke sertifikat CA root Amazon di perangkat klien.
- Ganti *`~/certs/device.pem.crt`* dengan jalur ke sertifikat perangkat pada perangkat klien.
- Ganti *`~/certs/private.pem.key`* dengan jalur menuju file kunci pribadi pada perangkat klien.
- Ganti *`us-east-1`* dengan Wilayah AWS di mana perangkat klien dan perangkat inti Anda beroperasi.

```
python3 basic_discovery.py \\
  --thing_name MyClientDevice1 \\
  --topic 'clients/MyClientDevice1/hello/world' \\
  --message 'Hello World!' \\
  --ca_file ~/certs/AmazonRootCA1.pem \\
  --cert ~/certs/device.pem.crt \\
  --key ~/certs/private.pem.key \\
  --region us-east-1 \\
  --verbosity Warn
```

Aplikasi sampel penemuan mengirimkan pesan 10 kali dan terputus. Aplikasi ini juga berlangganan topik yang sama di mana ia menerbitkan pesan. Jika output menunjukkan bahwa aplikasi itu menerima pesan MQTT pada topik, perangkat klien dapat berhasil berkomunikasi dengan perangkat inti.

```
Performing greengrass discovery...
awsiot.greengrass_discovery.DiscoverResponse(gg_groups=[awsiot.greengrass_discovery.GGGroup
coreDevice-MyGreengrassCore',
 cores=[awsiot.greengrass_discovery.GGCore(thing_arn='arn:aws:iot:us-
east-1:123456789012:thing/MyGreengrassCore',
 connectivity=[awsiot.greengrass_discovery.ConnectivityInfo(id='203.0.113.0',
 host_address='203.0.113.0', metadata='', port=8883)])],
 certificate_authorities=['-----BEGIN CERTIFICATE-----\
MIICiT...EXAMPLE=\
-----END CERTIFICATE-----\
'])])
Trying core arn:aws:iot:us-east-1:123456789012:thing/MyGreengrassCore at host
 203.0.113.0 port 8883
Connected!
Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 0}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 0}'
Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 1}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 1}'

...

Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 9}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 9}'
```

Jika aplikasi mengeluarkan kesalahan, lihat Pemecahan masalah penemuan Greengrass.

Anda juga dapat melihat log Greengrass pada perangkat inti untuk memverifikasi apakah perangkat klien berhasil menghubungkan dan mengirim pesan. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

4. Verifikasi bahwa jembatan MQTT merelai pesan dari perangkat klien ke AWS IoT Core. Anda dapat menggunakan klien tes MQTT di konsol AWS IoT Core untuk berlangganan filter topik MQTT. Lakukan hal-hal berikut:

   a. Navigasikan ke konsol AWS IoT tersebut.

   b. Di menu navigasi sebelah kiri, di bawah Pengujian pilih klien uji MQTT.

   c. Pada tab Berlangganan topik, untuk Filter topik, masukkan `clients/+/hello/world` untuk berlangganan pesan perangkat klien dari perangkat inti.

   d. Pilih Langganan.

   e. Jalankan aplikasi publish/subscribe pada perangkat klien lagi.

      Klien tes MQTT menampilkan pesan yang Anda kirim dari perangkat klien pada topik yang cocok dengan filter topik ini.

Langkah 4: Kembangkan komponen yang berkomunikasi dengan perangkat klien

Anda dapat mengembangkan komponen Greengrass yang berkomunikasi dengan perangkat klien. Komponen menggunakan komunikasi antar proses (IPC) dan antarmuka publish/subscribe lokal untuk berkomunikasi di perangkat inti. Untuk berinteraksi dengan perangkat klien, konfigurasikan komponen jembatan MQTT untuk menyampaikan pesan antara perangkat klien dan antarmuka penerbitan/langganan lokal.

Pada bagian ini, Anda memperbarui komponen jembatan MQTT untuk merelai pesan dari perangkat klien untuk lokal antarmuka publish/subscribe. Kemudian, Anda mengembangkan komponen yang berlangganan pesan ini dan mencetak pesan ketika menerimanya.

Untuk mengembangkan komponen yang berkomunikasi dengan perangkat klien

1. Revisi deployment ke perangkat inti dan konfigurasikan komponen jembatan MQTT untuk menyampaikan pesan dari perangkat klien ke antarmuka lokal. Lakukan hal-hal berikut:

   a. Navigasikan ke konsol AWS IoT Greengrass tersebut.

b. Di menu navigasi kiri, pilih Perangkat inti.

c. Pada halaman Perangkat inti, pilih perangkat inti yang Anda gunakan untuk tutorial ini.

d. Pada halaman detail perangkat inti, pilih tab Perangkat klien.

e. Pada tab Perangkat klien, pilih Konfigurasikan penemuan cloud.

   Halaman Konfigurasikan penemuan perangkat inti terbuka. Pada halaman ini, Anda dapat mengubah atau mengonfigurasi komponen perangkat klien mana yang di-deploy ke perangkat inti.

f. Di Langkah 3, untuk komponen aws.greengrass.clientdevices.mqtt.Bridge, pilih Edit konfigurasi.

g. Di modal Edit konfigurasi untuk komponen jembatan MQTT, konfigurasikan pemetaan topik yang menyampaikan pesan MQTT dari perangkat klien ke antarmuka publish/subscribe lokal. Lakukan hal-hal berikut:

   i. Di bawah Konfigurasi, di blok kode Konfigurasi yang akan digabungkan, masukkan konfigurasi berikut. Konfigurasi ini menentukan untuk menyampaikan pesan MQTT pada topik yang cocok dengan topik filter `clients/+/hello/world` dari perangkat klien ke layanan cloud AWS IoT Core dan broker publish/subscribe Greengrass lokal.

```
{
  "mqttTopicMapping": {
    "HelloWorldIotCoreMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "HelloWorldPubsubMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "Pubsub"
    }
  }
}
```

   Untuk informasi lebih lanjut, lihat konfigurasi komponen jembatan MQTT.

   ii. Pilih Konfirmasi.

h. Pilih Tinjau dan deploy untuk meninjau deployment yang dibuat oleh halaman ini untuk Anda.

i. Pada halaman Tinjauan, pilih Deploy untuk memulai deployment ke perangkat inti.

j. Untuk memverifikasi bahwa deployment berhasil, periksa status deployment, dan periksa log pada perangkat inti. Untuk memeriksa status deployment pada perangkat inti, Anda dapat memilih Target pada Gambaran Umum deployment. Untuk informasi selengkapnya, lihat hal berikut:

- Periksa status deployment
- Memantau AWS IoT Greengrass log

2. Kembangkan dan deploy komponen Greengrass yang berlangganan pesan Hello World dari perangkat klien. Lakukan hal-hal berikut:

a. Buat folder untuk resep dan artefak pada perangkat inti.

Linux or Unix

```
mkdir recipes
mkdir -p artifacts/com.example.clientdevices.MyHelloWorldSubscriber/1.0.0
```

Windows Command Prompt (CMD)

```
mkdir recipes
mkdir artifacts\com.example.clientdevices.MyHelloWorldSubscriber\1.0.0
```

PowerShell

```
mkdir recipes
mkdir artifacts\com.example.clientdevices.MyHelloWorldSubscriber\1.0.0
```

Important

Anda harus menggunakan format berikut untuk jalur folder artifact. Sertakan nama dan versi komponen yang Anda tentukan dalam resep.

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

b. Gunakan editor teks untuk membuat resep komponen dengan konten berikut. Resep ini menentukan untuk menginstal AWS IoT Device SDK v2 untuk Python dan menjalankan skrip yang berlangganan topik dan mencetak pesan.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   ```
   nano recipes/com.example.clientdevices.MyHelloWorldSubscriber-1.0.0.json
   ```

   Salin resep berikut ke dalam file.

   ```
   {
     "RecipeFormatVersion": "2020-01-25",
     "ComponentName": "com.example.clientdevices.MyHelloWorldSubscriber",
     "ComponentVersion": "1.0.0",
     "ComponentDescription": "A component that subscribes to Hello World messages
    from client devices.",
     "ComponentPublisher": "Amazon",
     "ComponentConfiguration": {
       "DefaultConfiguration": {
         "accessControl": {
           "aws.greengrass.ipc.pubsub": {
             "com.example.clientdevices.MyHelloWorldSubscriber:pubsub:1": {
               "policyDescription": "Allows access to subscribe to all topics.",
               "operations": [
                 "aws.greengrass#SubscribeToTopic"
               ],
               "resources": [
                 "*"
               ]
             }
           }
         }
       }
     },
     "Manifests": [
       {
         "Platform": {
           "os": "linux"
         },
         "Lifecycle": {
           "install": "python3 -m pip install --user awsiotsdk",
   ```

```
        "run": "python3 -u {artifacts:path}/hello_world_subscriber.py"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/hello_world_subscriber.py"
      }
    }
  ]
}
```

c. Gunakan editor teks untuk membuat artefak skrip Python bernama `hello_world_subscriber.py` dengan konten berikut. Aplikasi ini menggunakan layanan IPC publish/subscribe untuk berlangganan topik `clients/+/hello/world` dan pesan cetak yang diterimanya.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano artifacts/com.example.clientdevices.MyHelloWorldSubscriber/1.0.0/
hello_world_subscriber.py
```

   Salin kode Python berikut ke dalam file.

```
import sys
import time
import traceback

from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2

CLIENT_DEVICE_HELLO_WORLD_TOPIC = 'clients/+/hello/world'
TIMEOUT = 10


def on_hello_world_message(event):
    try:
        message = str(event.binary_message.message, 'utf-8')
        print('Received new message: %s' % message)
```

```
    except:
        traceback.print_exc()


try:
    ipc_client = GreengrassCoreIPCClientV2()

    # SubscribeToTopic returns a tuple with the response and the operation.
    _, operation = ipc_client.subscribe_to_topic(
        topic=CLIENT_DEVICE_HELLO_WORLD_TOPIC,
 on_stream_event=on_hello_world_message)
    print('Successfully subscribed to topic: %s' %
          CLIENT_DEVICE_HELLO_WORLD_TOPIC)

    # Keep the main thread alive, or the process will exit.
    try:
        while True:
            time.sleep(10)
    except InterruptedError:
        print('Subscribe interrupted.')

    operation.close()
except Exception:
    print('Exception occurred when using IPC.', file=sys.stderr)
    traceback.print_exc()
    exit(1)
```

> **Note**
>
> Komponen ini menggunakan klien IPC V2 di AWS IoT Device SDKv2 untuk Python untuk berkomunikasi dengan AWS IoT Greengrass perangkat lunak Core. Dibandingkan dengan klien IPC asli, klien IPC V2 mengurangi jumlah kode yang perlu Anda tulis untuk menggunakan IPC dalam komponen khusus.

d. Gunakn Greengrass CLI untuk men-deploy komponen.

Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
```

```
  --merge "com.example.clientdevices.MyHelloWorldSubscriber=1.0.0"
```

Windows Command Prompt (CMD)

```
C:\greengrass\v2/bin/greengrass-cli deployment create ^
  --recipeDir recipes ^
  --artifactDir artifacts ^
  --merge "com.example.clientdevices.MyHelloWorldSubscriber=1.0.0"
```

PowerShell

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

3. Lihat log komponen untuk memverifikasi bahwa komponen berhasil menginstal dan berlangganan topik.

   Linux or Unix

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

   PowerShell

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

   Anda dapat menjaga log feed tetap terbuka untuk memverifikasi bahwa inti perangkat menerima pesan.

4. Pada perangkat klien, jalankan sampel aplikasi penemuan Greengrass lagi untuk mengirim pesan ke perangkat inti.

   ```
   python3 basic_discovery.py \\
     --thing_name MyClientDevice1 \\
     --topic 'clients/MyClientDevice1/hello/world' \\
     --message 'Hello World!' \\
     --ca_file ~/certs/AmazonRootCA1.pem \\
   ```

```
  --cert ~/certs/device.pem.crt \\
  --key ~/certs/private.pem.key \\
  --region us-east-1 \\
  --verbosity Warn
```

5. Lihat log komponen lagi untuk memverifikasi bahwa komponen menerima dan mencetak pesan dari perangkat klien.

   Linux or Unix

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

   PowerShell

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

Langkah 5: Kembangkan komponen yang berinteraksi dengan bayangan perangkat klien

Anda dapat mengembangkan komponen Greengrass yang berinteraksi dengan bayangan perangkat perangkat klien. AWS IoT Bayangan adalah dokumen JSON yang menyimpan informasi status saat ini atau yang diinginkan untuk suatu AWS IoT hal, seperti perangkat klien. Komponen khusus dapat mengakses bayangan perangkat klien untuk mengelola statusnya, bahkan ketika perangkat klien tidak terhubung. AWS IoT Setiap AWS IoT benda memiliki bayangan yang tidak disebutkan namanya, dan Anda juga dapat membuat beberapa bayangan bernama untuk setiap hal.

Di bagian ini, Anda menerapkan komponen shadow manager untuk mengelola bayangan pada perangkat inti. Anda juga memperbarui komponen jembatan MQTT untuk menyampaikan pesan bayangan antara perangkat klien dan komponen shadow manager. Kemudian, Anda mengembangkan komponen yang memperbarui bayangan perangkat klien, dan Anda menjalankan aplikasi sampel pada perangkat klien yang merespons pembaruan bayangan dari komponen. Komponen ini mewakili aplikasi manajemen cahaya pintar, di mana perangkat inti mengelola status warna lampu pintar yang terhubung dengannya sebagai perangkat klien.

Untuk mengembangkan komponen yang berinteraksi dengan bayangan perangkat klien

1. Merevisi penerapan ke perangkat inti untuk menyebarkan komponen pengelola bayangan dan mengonfigurasi komponen jembatan MQTT untuk menyampaikan pesan bayangan antara perangkat klien dan penerbitan/langganan lokal, tempat manajer bayangan berkomunikasi. Lakukan hal-hal berikut:

   a. Navigasikan ke konsol AWS IoT Greengrass tersebut.

   b. Di menu navigasi kiri, pilih Perangkat inti.

   c. Pada halaman Perangkat inti, pilih perangkat inti yang Anda gunakan untuk tutorial ini.

   d. Pada halaman detail perangkat inti, pilih tab Perangkat klien.

   e. Pada tab Perangkat klien, pilih Konfigurasikan penemuan cloud.

      Halaman Konfigurasikan penemuan perangkat inti terbuka. Pada halaman ini, Anda dapat mengubah atau mengonfigurasi komponen perangkat klien mana yang di-deploy ke perangkat inti.

   f. Di Langkah 3, untuk komponen aws.greengrass.clientdevices.mqtt.Bridge, pilih Edit konfigurasi.

   g. Dalam modal konfigurasi Edit untuk komponen jembatan MQTT, konfigurasikan pemetaan topik yang menyampaikan pesan MQTT pada topik bayangan perangkat antara perangkat klien dan antarmuka penerbitan/langganan lokal. Anda juga mengonfirmasi bahwa penerapan menentukan versi jembatan MQTT yang kompatibel. Dukungan bayangan perangkat klien memerlukan jembatan MQTT v2.2.0 atau yang lebih baru. Lakukan hal-hal berikut:

      i. Untuk versi Komponen, pilih versi 2.2.0 atau yang lebih baru.

      ii. Di bawah Konfigurasi, di blok kode Konfigurasi yang akan digabungkan, masukkan konfigurasi berikut. Konfigurasi ini menentukan untuk menyampaikan pesan MQTT pada topik bayangan.

```
{
  "mqttTopicMapping": {
    "HelloWorldIotCoreMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "HelloWorldPubsubMapping": {
```

```
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Untuk informasi lebih lanjut, lihat konfigurasi komponen jembatan MQTT.

iii. Pilih Konfirmasi.

h. Pada Langkah 3, pilih aws.greengrass.ShadowManagerkomponen untuk menerapkannya.

i. Pilih Tinjau dan deploy untuk meninjau deployment yang dibuat oleh halaman ini untuk Anda.

j. Pada halaman Tinjauan, pilih Deploy untuk memulai deployment ke perangkat inti.

k. Untuk memverifikasi bahwa deployment berhasil, periksa status deployment, dan periksa log pada perangkat inti. Untuk memeriksa status deployment pada perangkat inti, Anda dapat memilih Target pada Gambaran Umum deployment. Untuk informasi selengkapnya, lihat hal berikut:

- Periksa status deployment
- Memantau AWS IoT Greengrass log

2. Kembangkan dan terapkan komponen Greengrass yang mengelola perangkat klien cahaya pintar. Lakukan hal-hal berikut:

a. Buat folder artefak komponen pada perangkat inti.

Linux or Unix

```
mkdir -p artifacts/com.example.clientdevices.MySmartLightManager/1.0.0
```

Windows Command Prompt (CMD)

```
mkdir artifacts\com.example.clientdevices.MySmartLightManager\1.0.0
```

PowerShell

```
mkdir artifacts\com.example.clientdevices.MySmartLightManager\1.0.0
```

> **Important**
>
> Anda harus menggunakan format berikut untuk jalur folder artifact. Sertakan nama dan versi komponen yang Anda tentukan dalam resep.
>
> ```
> artifacts/componentName/componentVersion/
> ```

b. Gunakan editor teks untuk membuat resep komponen dengan konten berikut. Resep ini menentukan untuk menginstal AWS IoT Device SDK v2 untuk Python dan menjalankan skrip yang berinteraksi dengan bayangan perangkat klien cahaya pintar untuk mengelola warnanya.

Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano recipes/com.example.clientdevices.MySmartLightManager-1.0.0.json
```

Salin resep berikut ke dalam file.

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.clientdevices.MySmartLightManager",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that interacts with smart light client
 devices.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.Nucleus": {
      "VersionRequirement": "^2.6.0"
```

```
    },
    "aws.greengrass.ShadowManager": {
      "VersionRequirement": "^2.2.0"
    },
    "aws.greengrass.clientdevices.mqtt.Bridge": {
      "VersionRequirement": "^2.2.0"
    }
  },
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "smartLightDeviceNames": [],
      "accessControl": {
        "aws.greengrass.ShadowManager": {
          "com.example.clientdevices.MySmartLightManager:shadow:1": {
            "policyDescription": "Allows access to client devices' unnamed
shadows",
            "operations": [
              "aws.greengrass#GetThingShadow",
              "aws.greengrass#UpdateThingShadow"
            ],
            "resources": [
              "$aws/things/MyClientDevice*/shadow"
            ]
          }
        },
        "aws.greengrass.ipc.pubsub": {
          "com.example.clientdevices.MySmartLightManager:pubsub:1": {
            "policyDescription": "Allows access to client devices' unnamed
shadow updates",
            "operations": [
              "aws.greengrass#SubscribeToTopic"
            ],
            "resources": [
              "$aws/things/+/shadow/update/accepted"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
```

```
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk",
        "run": "python3 -u {artifacts:path}/smart_light_manager.py"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/smart_light_manager.py"
      }
    }
  ]
}
```

c. Gunakan editor teks untuk membuat artefak skrip Python bernama `smart_light_manager.py` dengan konten berikut. Aplikasi ini menggunakan layanan shadow IPC untuk mendapatkan dan memperbarui bayangan perangkat klien dan layanan IPC penerbitan/berlangganan lokal untuk menerima pembaruan bayangan yang dilaporkan.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

```
nano artifacts/com.example.clientdevices.MySmartLightManager/1.0.0/
smart_light_manager.py
```

   Salin kode Python berikut ke dalam file.

```
import json
import random
import sys
import time
import traceback
from uuid import uuid4

from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
from awsiot.greengrasscoreipc.model import ResourceNotFoundError

SHADOW_COLOR_PROPERTY = 'color'
```

```
CONFIGURATION_CLIENT_DEVICE_NAMES = 'smartLightDeviceNames'
COLORS = ['red', 'orange', 'yellow', 'green', 'blue', 'purple']
SHADOW_UPDATE_TOPIC = '$aws/things/+/shadow/update/accepted'
SET_COLOR_INTERVAL = 15


class SmartLightDevice():
    def __init__(self, client_device_name: str, reported_color: str = None):
        self.name = client_device_name
        self.reported_color = reported_color
        self.desired_color = None


class SmartLightDeviceManager():
    def __init__(self, ipc_client: GreengrassCoreIPCClientV2):
        self.ipc_client = ipc_client
        self.devices = {}
        self.client_tokens = set()
        self.shadow_update_accepted_subscription_operation = None
        self.client_device_names_configuration_subscription_operation = None
        self.update_smart_light_device_list()

    def update_smart_light_device_list(self):
        # Update the device list from the component configuration.
        response = self.ipc_client.get_configuration(
            key_path=[CONFIGURATION_CLIENT_DEVICE_NAMES])
        # Identify the difference between the configuration and the currently
 tracked devices.
        current_device_names = self.devices.keys()
        updated_device_names =
 response.value[CONFIGURATION_CLIENT_DEVICE_NAMES]
        added_device_names = set(updated_device_names) -
 set(current_device_names)
        removed_device_names = set(current_device_names) -
 set(updated_device_names)
        # Stop tracking any smart light devices that are no longer in the
 configuration.
        for name in removed_device_names:
            print('Removing %s from smart light device manager' % name)
            self.devices.pop(name)
        # Start tracking any new smart light devices that are in the
 configuration.
        for name in added_device_names:
            print('Adding %s to smart light device manager' % name)
```

```
            device = SmartLightDevice(name)
            device.reported_color = self.get_device_reported_color(device)
            self.devices[name] = device
            print('Current color for %s is %s' % (name,
device.reported_color))

    def get_device_reported_color(self, smart_light_device):
        try:
            response = self.ipc_client.get_thing_shadow(
                thing_name=smart_light_device.name, shadow_name='')
            shadow = json.loads(str(response.payload, 'utf-8'))
            if 'reported' in shadow['state']:
                return shadow['state']['reported'].get(SHADOW_COLOR_PROPERTY)
            return None
        except ResourceNotFoundError:
            return None

    def request_device_color_change(self, smart_light_device, color):
        # Generate and track a client token for the request.
        client_token = str(uuid4())
        self.client_tokens.add(client_token)
        # Create a shadow payload, which must be a blob.
        payload_json = {
            'state': {
                'desired': {
                    SHADOW_COLOR_PROPERTY: color
                }
            },
            'clientToken': client_token
        }
        payload = bytes(json.dumps(payload_json), 'utf-8')
        self.ipc_client.update_thing_shadow(
            thing_name=smart_light_device.name, shadow_name='',
payload=payload)
        smart_light_device.desired_color = color

    def subscribe_to_shadow_update_accepted_events(self):
        if self.shadow_update_accepted_subscription_operation == None:
            # SubscribeToTopic returns a tuple with the response and the
operation.
            _, self.shadow_update_accepted_subscription_operation =
self.ipc_client.subscribe_to_topic(
                topic=SHADOW_UPDATE_TOPIC,
on_stream_event=self.on_shadow_update_accepted_event)
```

```
            print('Successfully subscribed to shadow update accepted topic')

    def close_shadow_update_accepted_subscription(self):
        if self.shadow_update_accepted_subscription_operation is not None:
            self.shadow_update_accepted_subscription_operation.close()

    def on_shadow_update_accepted_event(self, event):
        try:
            message = str(event.binary_message.message, 'utf-8')
            accepted_payload = json.loads(message)
            # Check for reported states from smart light devices and ignore
desired states from components.
            if 'reported' in accepted_payload['state']:
                # Process this update only if it uses a client token created by
this component.
                client_token = accepted_payload.get('clientToken')
                if client_token is not None and client_token in
self.client_tokens:
                    self.client_tokens.remove(client_token)
                    shadow_state = accepted_payload['state']['reported']
                    if SHADOW_COLOR_PROPERTY in shadow_state:
                        reported_color = shadow_state[SHADOW_COLOR_PROPERTY]
                        topic = event.binary_message.context.topic
                        client_device_name = topic.split('/')[2]
                        if client_device_name in self.devices:
                            # Set the reported color for the smart light
device.
                            self.devices[client_device_name].reported_color =
reported_color
                            print(
                                'Received shadow update confirmation from
client device: %s' % client_device_name)
                    else:
                        print("Shadow update doesn't specify color")
        except:
            traceback.print_exc()

    def subscribe_to_client_device_name_configuration_updates(self):
        if self.client_device_names_configuration_subscription_operation ==
None:
            # SubscribeToConfigurationUpdate returns a tuple with the response
and the operation.
            _, self.client_device_names_configuration_subscription_operation =
self.ipc_client.subscribe_to_configuration_update(
```

```
                key_path=[CONFIGURATION_CLIENT_DEVICE_NAMES],
 on_stream_event=self.on_client_device_names_configuration_update_event)
            print(
                'Successfully subscribed to configuration updates for smart
 light device names')

    def close_client_device_names_configuration_subscription(self):
        if self.client_device_names_configuration_subscription_operation is not
 None:

 self.client_device_names_configuration_subscription_operation.close()

    def on_client_device_names_configuration_update_event(self, event):
        try:
            if CONFIGURATION_CLIENT_DEVICE_NAMES in
 event.configuration_update_event.key_path:
                print('Received configuration update for list of client
 devices')
                self.update_smart_light_device_list()
        except:
            traceback.print_exc()


def choose_random_color():
    return random.choice(COLORS)

def main():
    try:
        # Create an IPC client and a smart light device manager.
        ipc_client = GreengrassCoreIPCClientV2()
        smart_light_manager = SmartLightDeviceManager(ipc_client)
        smart_light_manager.subscribe_to_shadow_update_accepted_events()

 smart_light_manager.subscribe_to_client_device_name_configuration_updates()
        try:
            # Keep the main thread alive, or the process will exit.
            while True:
                # Set each smart light device to a random color at a regular
 interval.
                for device_name in smart_light_manager.devices:
                    device = smart_light_manager.devices[device_name]
                    desired_color = choose_random_color()
                    print('Chose random color (%s) for %s' %
                          (desired_color, device_name))
```

```
                    if desired_color == device.desired_color:
                        print('Desired color for %s is already %s' %
                              (device_name, desired_color))
                    elif desired_color == device.reported_color:
                        print('Reported color for %s is already %s' %
                              (device_name, desired_color))
                    else:
                        smart_light_manager.request_device_color_change(
                            device, desired_color)
                        print('Requested color change for %s to %s' %
                              (device_name, desired_color))
                time.sleep(SET_COLOR_INTERVAL)
        except InterruptedError:
            print('Application interrupted')
        smart_light_manager.close_shadow_update_accepted_subscription()

 smart_light_manager.close_client_device_names_configuration_subscription()
    except Exception:
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)


if __name__ == '__main__':
    main()
```

Aplikasi Python ini melakukan hal berikut:

- Membaca konfigurasi komponen untuk mendapatkan daftar perangkat klien cahaya pintar untuk dikelola.
- Berlangganan pemberitahuan pembaruan konfigurasi menggunakan operasi SubscribeToConfigurationUpdate IPC. Perangkat lunak AWS IoT Greengrass Core mengirimkan pemberitahuan setiap kali konfigurasi komponen berubah. Saat komponen menerima pemberitahuan pembaruan konfigurasi, komponen akan memperbarui daftar perangkat klien cahaya pintar yang dikelolanya.
- Mendapat bayangan setiap perangkat klien cahaya pintar untuk mendapatkan status warna awalnya.
- Mengatur warna setiap perangkat klien cahaya pintar ke warna acak setiap 15 detik. Komponen memperbarui bayangan benda perangkat klien untuk mengubah warnanya. Operasi ini mengirimkan peristiwa delta bayangan ke perangkat klien melalui MQTT.

- Berlangganan pembaruan bayangan pesan yang diterima pada antarmuka penerbitan/berlangganan lokal menggunakan operasi IPC. SubscribeToTopic Komponen ini menerima pesan-pesan ini untuk melacak warna setiap perangkat klien cahaya pintar. Ketika perangkat klien cahaya pintar menerima pembaruan bayangan, ia mengirimkan pesan MQTT untuk mengonfirmasi bahwa ia menerima pembaruan. Jembatan MQTT menyampaikan pesan ini ke antarmuka penerbitan/berlangganan lokal.

d. Gunakn Greengrass CLI untuk men-deploy komponen. Saat menerapkan komponen ini, Anda menentukan daftar perangkat kliensmartLightDeviceNames, yang bayangannya dikelola. Ganti *MyClientDevice1* dengan nama benda perangkat klien.

Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.clientdevices.MySmartLightManager=1.0.0" \
  --update-config '{
    "com.example.clientdevices.MySmartLightManager": {
      "MERGE": {
        "smartLightDeviceNames": [
          "MyClientDevice1"
        ]
      }
    }
  }'
```

Windows Command Prompt (CMD)

```
C:\greengrass\v2/bin/greengrass-cli deployment create ^
  --recipeDir recipes ^
  --artifactDir artifacts ^
  --merge "com.example.clientdevices.MySmartLightManager=1.0.0" ^
  --update-config '{"com.example.clientdevices.MySmartLightManager":
{"MERGE":{"smartLightDeviceNames":["MyClientDevice1"]}}}'
```

PowerShell

```
C:\greengrass\v2/bin/greengrass-cli deployment create `
  --recipeDir recipes `
  --artifactDir artifacts `
```

```
  --merge "com.example.clientdevices.MySmartLightManager=1.0.0" `
  --update-config '{
    "com.example.clientdevices.MySmartLightManager": {
      "MERGE": {
        "smartLightDeviceNames": [
          "MyClientDevice1"
        ]
      }
    }
  }'
```

3. Lihat log komponen untuk memverifikasi bahwa komponen menginstal dan berjalan dengan sukses.

   Linux or Unix

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

   PowerShell

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

   Komponen mengirimkan permintaan untuk mengubah warna perangkat klien cahaya pintar. Manajer bayangan menerima permintaan dan menetapkan `desired` status bayangan. Namun, perangkat klien cahaya pintar belum berjalan, sehingga `reported` status bayangan tidak berubah. Log komponen menyertakan pesan-pesan berikut.

   ```
   2022-07-07T03:49:24.908Z [INFO] (Copier)
    com.example.clientdevices.MySmartLightManager: stdout. Chose random color (blue)
    for MyClientDevice1.
    {scriptName=services.com.example.clientdevices.MySmartLightManager.lifecycle.Run,
    serviceName=com.example.clientdevices.MySmartLightManager, currentState=RUNNING}
   2022-07-07T03:49:24.912Z [INFO] (Copier)
    com.example.clientdevices.MySmartLightManager: stdout.
    Requested color change for MyClientDevice1 to blue.
    {scriptName=services.com.example.clientdevices.MySmartLightManager.lifecycle.Run,
    serviceName=com.example.clientdevices.MySmartLightManager, currentState=RUNNING}
   ```

Anda dapat membiarkan umpan log tetap terbuka untuk melihat kapan komponen mencetak pesan.

4. Unduh dan jalankan contoh aplikasi yang menggunakan penemuan Greengrass dan berlangganan pembaruan bayangan perangkat. Di perangkat klien, lakukan hal berikut:

   a. Ubah ke folder contoh di v2 for Python AWS IoT Device SDK. Aplikasi sampel ini menggunakan modul parsing baris perintah di folder sampel.

   ```
   cd aws-iot-device-sdk-python-v2/samples
   ```

   b. Gunakan editor teks untuk membuat skrip Python bernama `basic_discovery_shadow.py` dengan konten berikut. Aplikasi ini menggunakan penemuan Greengrass dan bayangan untuk menjaga properti tetap sinkron antara perangkat klien dan perangkat inti.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   ```
   nano basic_discovery_shadow.py
   ```

   Salin kode Python berikut ke dalam file.

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0.

   from awscrt import io
   from awscrt import mqtt
   from awsiot import iotshadow
   from awsiot.greengrass_discovery import DiscoveryClient
   from awsiot import mqtt_connection_builder
   from concurrent.futures import Future
   import sys
   import threading
   import traceback
   from uuid import uuid4

   # Parse arguments
   import utils.command_line_utils;
   cmdUtils = utils.command_line_utils.CommandLineUtils("Basic Discovery -
    Greengrass discovery example with device shadows.")
   ```

```
cmdUtils.add_common_mqtt_commands()
cmdUtils.add_common_topic_message_commands()
cmdUtils.add_common_logging_commands()
cmdUtils.register_command("key", "<path>", "Path to your key in PEM format.",
 True, str)
cmdUtils.register_command("cert", "<path>", "Path to your client certificate in
 PEM format.", True, str)
cmdUtils.remove_command("endpoint")
cmdUtils.register_command("thing_name", "<str>", "The name assigned to your IoT
 Thing", required=True)
cmdUtils.register_command("region", "<str>", "The region to connect through.",
 required=True)
cmdUtils.register_command("shadow_property", "<str>", "The name of the shadow
 property you want to change (optional, default='color'", default="color")
# Needs to be called so the command utils parse the commands
cmdUtils.get_args()

# Using globals to simplify sample code
is_sample_done = threading.Event()
mqtt_connection = None
shadow_thing_name = cmdUtils.get_command_required("thing_name")
shadow_property = cmdUtils.get_command("shadow_property")

SHADOW_VALUE_DEFAULT = "off"

class LockedData:
    def __init__(self):
        self.lock = threading.Lock()
        self.shadow_value = None
        self.disconnect_called = False
        self.request_tokens = set()

locked_data = LockedData()


def on_connection_interupted(connection, error, **kwargs):
    print('connection interrupted with error {}'.format(error))


def on_connection_resumed(connection, return_code, session_present, **kwargs):
    print('connection resumed with return code {}, session present
 {}'.format(return_code, session_present))

# Try IoT endpoints until we find one that works
```

```
def try_iot_endpoints():
    for gg_group in discover_response.gg_groups:
        for gg_core in gg_group.cores:
            for connectivity_info in gg_core.connectivity:
                try:
                    print('Trying core {} at host {} port
 {}'.format(gg_core.thing_arn, connectivity_info.host_address,
 connectivity_info.port))
                    mqtt_connection = mqtt_connection_builder.mtls_from_path(
                        endpoint=connectivity_info.host_address,
                        port=connectivity_info.port,
                        cert_filepath=cmdUtils.get_command_required("cert"),
                        pri_key_filepath=cmdUtils.get_command_required("key"),

 ca_bytes=gg_group.certificate_authorities[0].encode('utf-8'),
                        on_connection_interrupted=on_connection_interupted,
                        on_connection_resumed=on_connection_resumed,
                        client_id=cmdUtils.get_command_required("thing_name"),
                        clean_session=False,
                        keep_alive_secs=30)

                    connect_future = mqtt_connection.connect()
                    connect_future.result()
                    print('Connected!')
                    return mqtt_connection

                except Exception as e:
                    print('Connection failed with exception {}'.format(e))
                    continue

    exit('All connection attempts failed')

# Function for gracefully quitting this sample
def exit(msg_or_exception):
    if isinstance(msg_or_exception, Exception):
        print("Exiting sample due to exception.")
        traceback.print_exception(msg_or_exception.__class__, msg_or_exception,
 sys.exc_info()[2])
    else:
        print("Exiting sample:", msg_or_exception)

    with locked_data.lock:
        if not locked_data.disconnect_called:
            print("Disconnecting...")
```

```
            locked_data.disconnect_called = True
            future = mqtt_connection.disconnect()
            future.add_done_callback(on_disconnected)

def on_disconnected(disconnect_future):
    # type: (Future) -> None
    print("Disconnected.")

    # Signal that sample is finished
    is_sample_done.set()

def on_get_shadow_accepted(response):
    # type: (iotshadow.GetShadowResponse) -> None
    try:
        with locked_data.lock:
            # check that this is a response to a request from this session
            try:
                locked_data.request_tokens.remove(response.client_token)
            except KeyError:
                return

            print("Finished getting initial shadow state.")
            if locked_data.shadow_value is not None:
                print("  Ignoring initial query because a delta event has
 already been received.")
                return

        if response.state:
            if response.state.delta:
                value = response.state.delta.get(shadow_property)
                if value:
                    print("  Shadow contains delta value '{}'.".format(value))
                    change_shadow_value(value)
                    return

            if response.state.reported:
                value = response.state.reported.get(shadow_property)
                if value:
                    print("  Shadow contains reported value
 '{}'.".format(value))

 set_local_value_due_to_initial_query(response.state.reported[shadow_property])
                    return
```

```
        print("  Shadow document lacks '{}' property. Setting
 defaults...".format(shadow_property))
        change_shadow_value(SHADOW_VALUE_DEFAULT)
        return

    except Exception as e:
        exit(e)

def on_get_shadow_rejected(error):
    # type: (iotshadow.ErrorResponse) -> None
    try:
        # check that this is a response to a request from this session
        with locked_data.lock:
            try:
                locked_data.request_tokens.remove(error.client_token)
            except KeyError:
                return

        if error.code == 404:
            print("Thing has no shadow document. Creating with defaults...")
            change_shadow_value(SHADOW_VALUE_DEFAULT)
        else:
            exit("Get request was rejected. code:{} message:'{}'".format(
                error.code, error.message))

    except Exception as e:
        exit(e)

def on_shadow_delta_updated(delta):
    # type: (iotshadow.ShadowDeltaUpdatedEvent) -> None
    try:
        print("Received shadow delta event.")
        if delta.state and (shadow_property in delta.state):
            value = delta.state[shadow_property]
            if value is None:
                print("  Delta reports that '{}' was deleted. Resetting
 defaults...".format(shadow_property))
                change_shadow_value(SHADOW_VALUE_DEFAULT)
                return
            else:
                print("  Delta reports that desired value is '{}'. Changing
 local value...".format(value))
                if (delta.client_token is not None):
                    print ("  ClientToken is: " + delta.client_token)
```

```
                change_shadow_value(value, delta.client_token)
        else:
            print("  Delta did not report a change in
 '{}'".format(shadow_property))

    except Exception as e:
        exit(e)

def on_publish_update_shadow(future):
    #type: (Future) -> None
    try:
        future.result()
        print("Update request published.")
    except Exception as e:
        print("Failed to publish update request.")
        exit(e)

def on_update_shadow_accepted(response):
    # type: (iotshadow.UpdateShadowResponse) -> None
    try:
        # check that this is a response to a request from this session
        with locked_data.lock:
            try:
                locked_data.request_tokens.remove(response.client_token)
            except KeyError:
                return

        try:
            if response.state.reported != None:
                if shadow_property in response.state.reported:
                    print("Finished updating reported shadow value to
 '{}'.".format(response.state.reported[shadow_property])) # type: ignore
                else:
                    print ("Could not find shadow property with name:
 '{}'.".format(shadow_property)) # type: ignore
            else:
                print("Shadow states cleared.") # when the shadow states are
 cleared, reported and desired are set to None
        except:
            exit("Updated shadow is missing the target property")

    except Exception as e:
        exit(e)
```

```
def on_update_shadow_rejected(error):
    # type: (iotshadow.ErrorResponse) -> None
    try:
        # check that this is a response to a request from this session
        with locked_data.lock:
            try:
                locked_data.request_tokens.remove(error.client_token)
            except KeyError:
                return

        exit("Update request was rejected. code:{} message:'{}'".format(
            error.code, error.message))

    except Exception as e:
        exit(e)

def set_local_value_due_to_initial_query(reported_value):
    with locked_data.lock:
        locked_data.shadow_value = reported_value

def change_shadow_value(value, token=None):
    with locked_data.lock:
        if locked_data.shadow_value == value:
            print("Local value is already '{}'.".format(value))
            return

        print("Changed local shadow value to '{}'.".format(value))
        locked_data.shadow_value = value

        print("Updating reported shadow value to '{}'...".format(value))

        reuse_token = token is not None
        # use a unique token so we can correlate this "request" message to
        # any "response" messages received on the /accepted and /rejected
 topics
        if not reuse_token:
            token = str(uuid4())

        # if the value is "clear shadow" then send a UpdateShadowRequest with
 None
        # for both reported and desired to clear the shadow document
 completely.
        if value == "clear_shadow":
```

```
            tmp_state = iotshadow.ShadowState(reported=None, desired=None,
 reported_is_nullable=True, desired_is_nullable=True)
            request = iotshadow.UpdateShadowRequest(
                thing_name=shadow_thing_name,
                state=tmp_state,
                client_token=token,
            )
        # Otherwise, send a normal update request
        else:
            # if the value is "none" then set it to a Python none object to
            # clear the individual shadow property
            if value == "none":
                value = None

            request = iotshadow.UpdateShadowRequest(
            thing_name=shadow_thing_name,
            state=iotshadow.ShadowState(
                reported={ shadow_property: value }
                ),
                client_token=token,
            )

        future = shadow_client.publish_update_shadow(request,
 mqtt.QoS.AT_LEAST_ONCE)

        if not reuse_token:
            locked_data.request_tokens.add(token)

        future.add_done_callback(on_publish_update_shadow)


if __name__ == '__main__':
    tls_options =
 io.TlsContextOptions.create_client_with_mtls_from_path(cmdUtils.get_command_required("
 cmdUtils.get_command_required("key"))
    if cmdUtils.get_command(cmdUtils.m_cmd_ca_file):
        tls_options.override_default_trust_store_from_path(None,
 cmdUtils.get_command(cmdUtils.m_cmd_ca_file))
    tls_context = io.ClientTlsContext(tls_options)

    socket_options = io.SocketOptions()

    print('Performing greengrass discovery...')
```

```
   discovery_client =
DiscoveryClient(io.ClientBootstrap.get_or_create_static_default(),
socket_options, tls_context, cmdUtils.get_command_required("region"))
   resp_future =
discovery_client.discover(cmdUtils.get_command_required("thing_name"))
   discover_response = resp_future.result()

   print(discover_response)
   if cmdUtils.get_command("print_discover_resp_only"):
       exit(0)

   mqtt_connection = try_iot_endpoints()
   shadow_client = iotshadow.IotShadowClient(mqtt_connection)

   try:
       # Subscribe to necessary topics.
       # Note that is **is** important to wait for "accepted/rejected"
subscriptions
       # to succeed before publishing the corresponding "request".
       print("Subscribing to Update responses...")
       update_accepted_subscribed_future, _ =
shadow_client.subscribe_to_update_shadow_accepted(

request=iotshadow.UpdateShadowSubscriptionRequest(thing_name=shadow_thing_name),
           qos=mqtt.QoS.AT_LEAST_ONCE,
           callback=on_update_shadow_accepted)

       update_rejected_subscribed_future, _ =
shadow_client.subscribe_to_update_shadow_rejected(

request=iotshadow.UpdateShadowSubscriptionRequest(thing_name=shadow_thing_name),
           qos=mqtt.QoS.AT_LEAST_ONCE,
           callback=on_update_shadow_rejected)

       # Wait for subscriptions to succeed
       update_accepted_subscribed_future.result()
       update_rejected_subscribed_future.result()

       print("Subscribing to Get responses...")
       get_accepted_subscribed_future, _ =
shadow_client.subscribe_to_get_shadow_accepted(

request=iotshadow.GetShadowSubscriptionRequest(thing_name=shadow_thing_name),
           qos=mqtt.QoS.AT_LEAST_ONCE,
```

```
            callback=on_get_shadow_accepted)

        get_rejected_subscribed_future, _ =
shadow_client.subscribe_to_get_shadow_rejected(

request=iotshadow.GetShadowSubscriptionRequest(thing_name=shadow_thing_name),
            qos=mqtt.QoS.AT_LEAST_ONCE,
            callback=on_get_shadow_rejected)

        # Wait for subscriptions to succeed
        get_accepted_subscribed_future.result()
        get_rejected_subscribed_future.result()

        print("Subscribing to Delta events...")
        delta_subscribed_future, _ =
shadow_client.subscribe_to_shadow_delta_updated_events(

request=iotshadow.ShadowDeltaUpdatedSubscriptionRequest(thing_name=shadow_thing_name),
            qos=mqtt.QoS.AT_LEAST_ONCE,
            callback=on_shadow_delta_updated)

        # Wait for subscription to succeed
        delta_subscribed_future.result()

        # The rest of the sample runs asynchronously.

        # Issue request for shadow's current state.
        # The response will be received by the on_get_accepted() callback
        print("Requesting current shadow state...")

        with locked_data.lock:
            # use a unique token so we can correlate this "request" message to
            # any "response" messages received on the /accepted and /rejected
topics
            token = str(uuid4())

            publish_get_future = shadow_client.publish_get_shadow(

request=iotshadow.GetShadowRequest(thing_name=shadow_thing_name,
client_token=token),
                qos=mqtt.QoS.AT_LEAST_ONCE)

            locked_data.request_tokens.add(token)
```

```
        # Ensure that publish succeeds
        publish_get_future.result()

    except Exception as e:
        exit(e)

    # Wait for the sample to finish (user types 'quit', or an error occurs)
    is_sample_done.wait()
```

Aplikasi Python ini melakukan hal berikut:

- Menggunakan penemuan Greengrass untuk menemukan dan terhubung ke perangkat inti.
- Meminta dokumen bayangan dari perangkat inti untuk mendapatkan status awal properti.
- Berlangganan peristiwa delta bayangan, yang dikirim perangkat inti ketika `desired` nilai properti berbeda dari nilainya`reported`. Ketika aplikasi menerima peristiwa delta bayangan, itu mengubah nilai properti dan mengirimkan pembaruan ke perangkat inti untuk menetapkan nilai baru sebagai `reported` nilainya.

Aplikasi ini menggabungkan penemuan Greengrass dan sampel bayangan dari v2. AWS IoT Device SDK

c. Jalankan aplikasi sampel. Aplikasi ini mengharapkan argumen yang menentukan nama benda perangkat klien, properti bayangan yang akan digunakan, dan sertifikat yang mengautentikasi dan mengamankan koneksi.

- Ganti *`MyClientDevice1`* dengan nama benda perangkat klien.
- Ganti *`~/certs/ AmazonRoot ca1.pem dengan`* jalur ke sertifikat CA root Amazon di perangkat klien.
- Ganti *`~/certs/device.pem.crt`* dengan jalur ke sertifikat perangkat pada perangkat klien.
- Ganti *`~/certs/private.pem.key`* dengan jalur menuju file kunci pribadi pada perangkat klien.
- Ganti *`us-east-1`* dengan Wilayah AWS di mana perangkat klien dan perangkat inti Anda beroperasi.

```
python3 basic_discovery_shadow.py \
  --thing_name MyClientDevice1 \
```

```
  --shadow_property color \
  --ca_file ~/certs/AmazonRootCA1.pem \
  --cert ~/certs/device.pem.crt \
  --key ~/certs/private.pem.key \
  --region us-east-1 \
  --verbosity Warn
```

Aplikasi sampel berlangganan topik bayangan dan menunggu untuk menerima peristiwa delta bayangan dari perangkat inti. Jika output menunjukkan bahwa aplikasi menerima dan merespons peristiwa delta bayangan, perangkat klien dapat berhasil berinteraksi dengan bayangannya pada perangkat inti.

```
Performing greengrass discovery...
awsiot.greengrass_discovery.DiscoverResponse(gg_groups=[awsiot.greengrass_discovery.GGG
coreDevice-MyGreengrassCore',
 cores=[awsiot.greengrass_discovery.GGCore(thing_arn='arn:aws:iot:us-
east-1:123456789012:thing/MyGreengrassCore',
 connectivity=[awsiot.greengrass_discovery.ConnectivityInfo(id='203.0.113.0',
 host_address='203.0.113.0', metadata='', port=8883)])],
 certificate_authorities=['-----BEGIN CERTIFICATE-----
\nMIICiT...EXAMPLE=\n-----END CERTIFICATE-----\n'])])
Trying core arn:aws:iot:us-east-1:123456789012:thing/MyGreengrassCore at host
 203.0.113.0 port 8883
Connected!
Subscribing to Update responses...
Subscribing to Get responses...
Subscribing to Delta events...
Requesting current shadow state...
Received shadow delta event.
  Delta reports that desired value is 'purple'. Changing local value...
  ClientToken is: 3dce4d3f-e336-41ac-aa4f-7882725f0033
Changed local shadow value to 'purple'.
Updating reported shadow value to 'purple'...
Update request published.
```

Jika aplikasi mengeluarkan kesalahan, lihat Pemecahan masalah penemuan Greengrass.

Anda juga dapat melihat log Greengrass pada perangkat inti untuk memverifikasi apakah perangkat klien berhasil menghubungkan dan mengirim pesan. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

5. Lihat log komponen lagi untuk memverifikasi bahwa komponen menerima konfirmasi pembaruan bayangan dari perangkat klien cahaya pintar.

   Linux or Unix

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

   PowerShell

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

   Komponen mencatat pesan untuk mengonfirmasi bahwa perangkat klien cahaya pintar berubah warnanya.

   ```
   2022-07-07T03:49:24.908Z [INFO] (Copier)
    com.example.clientdevices.MySmartLightManager: stdout. Chose random color (blue)
    for MyClientDevice1.
    {scriptName=services.com.example.clientdevices.MySmartLightManager.lifecycle.Run,
    serviceName=com.example.clientdevices.MySmartLightManager, currentState=RUNNING}
   2022-07-07T03:49:24.912Z [INFO] (Copier)
    com.example.clientdevices.MySmartLightManager: stdout.
    Requested color change for MyClientDevice1 to blue.
    {scriptName=services.com.example.clientdevices.MySmartLightManager.lifecycle.Run,
    serviceName=com.example.clientdevices.MySmartLightManager, currentState=RUNNING}
   2022-07-07T03:49:24.959Z [INFO] (Copier)
    com.example.clientdevices.MySmartLightManager: stdout. Received
    shadow update confirmation from client device: MyClientDevice1.
    {scriptName=services.com.example.clientdevices.MySmartLightManager.lifecycle.Run,
    serviceName=com.example.clientdevices.MySmartLightManager, currentState=RUNNING}
   ```

> **Note**
>
> Bayangan perangkat klien disinkronkan antara perangkat inti dan perangkat klien. Namun, perangkat inti tidak menyinkronkan bayangan perangkat klien denganAWS IoT Core. Anda dapat menyinkronkan bayangan dengan AWS IoT Core untuk melihat atau mengubah status semua perangkat di armada Anda, misalnya. Untuk informasi selengkapnya tentang cara

mengonfigurasi komponen shadow manager untuk menyinkronkan bayanganAWS IoT Core, lihatSinkronkan bayangan perangkat lokal dengan AWS IoT Core.

Anda telah menyelesaikan tutorial ini. Perangkat klien terhubung ke perangkat inti, mengirim pesan MQTT ke dan komponen AWS IoT Core Greengrass, dan menerima pembaruan bayangan dari perangkat inti. Untuk informasi lebih lanjut tentang topik yang dibahas dalam tutorial ini, lihat berikut ini:

- Kaitkan perangkat klien
- Kelola titik akhir perangkat inti
- Uji komunikasi perangkat klien
- Greengrass discovery RESTful API
- Relai pesan MQTT antara perangkat klien dan AWS IoT Core
- Berinteraksilah dengan perangkat klien dalam komponen
- Berinteraksilah dengan bayangan perangkat
- Berinteraksi dengan dan menyinkronkan bayangan perangkat klien

Tutorial: Memulai dengan SageMaker Edge Manager

Important

SageMaker Edge Manager dihentikan pada 26 April 2024. Untuk informasi selengkapnya tentang melanjutkan penerapan model Anda ke perangkat edge, lihat SageMaker Edge Manager akhir masa pakai.

Amazon SageMaker Edge Manager adalah agen perangkat lunak yang berjalan pada perangkat edge. SageMaker Edge Manager menyediakan manajemen model untuk perangkat edge sehingga Anda dapat mengemas dan menggunakan model Amazon SageMaker Neo yang dikompilasi langsung di perangkat inti Greengrass. Dengan menggunakan SageMaker Edge Manager, Anda juga dapat mengambil sampel data input dan output model dari perangkat inti Anda, dan mengirim data tersebut ke AWS Cloud untuk pemantauan dan analisis. Untuk informasi selengkapnya tentang cara kerja SageMaker Edge Manager pada perangkat inti Greengrass, lihat. Gunakan Amazon SageMaker Edge Manager di perangkat inti Greengrass

Tutorial ini menunjukkan cara memulai menggunakan SageMaker Edge Manager dengan komponen sampel AWS yang disediakan pada perangkat inti yang ada. Komponen sampel ini menggunakan komponen SageMaker Edge Manager sebagai dependensi untuk menyebarkan agen Edge Manager, dan melakukan inferensi menggunakan model pra-terlatih yang dikompilasi menggunakan Neo. SageMaker Untuk informasi selengkapnya tentang agen SageMaker Edge Manager, lihat SageMaker Edge Manager di Panduan SageMaker Pengembang Amazon.

Untuk menyiapkan dan menggunakan agen SageMaker Edge Manager pada perangkat inti Greengrass yang ada AWS , berikan contoh kode yang dapat Anda gunakan untuk membuat inferensi sampel dan komponen model berikut.

- Klasifikasi gambar
  - `com.greengrass.SageMakerEdgeManager.ImageClassification`
  - `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`
- Deteksi objek
  - `com.greengrass.SageMakerEdgeManager.ObjectDetection`
  - `com.greengrass.SageMakerEdgeManager.ObjectDetection.Model`

Tutorial ini menunjukkan cara menerapkan komponen sampel dan agen SageMaker Edge Manager.

Topik

Prasyarat

Untuk menyelesaikan tutorial ini, Anda harus memenuhi prasyarat berikut:

- Perangkat inti Greengrass yang berjalan di Amazon Linux 2, platform Linux berbasis Debian (x86_64 atau Armv8), atau Windows (x86_64). Jika Anda tidak memilikinya, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2.
- Python 3.6 atau yang lebih baru, termasuk `pip` untuk versi Python Anda, diinstal pada perangkat inti anda.

- Waktu aktif OpenGL API GLX (`libgl1-mesa-glx`) yang terpasang pada perangkat inti Anda.
- Pengguna AWS Identity and Access Management (IAM) dengan izin administrator.
- Komputer pengembangan seperti Windows, Mac, atau UNIX dengan internet aktif yang memenuhi persyaratan berikut:
  - Python 3.6 atau yang lebih baru telah terinstal.
  - AWS CLI diinstal dan dikonfigurasi dengan kredensi pengguna administrator IAM Anda. Untuk informasi selengkapnya, lihat Menginstal AWS CLI dan Mengonfigurasi AWS CLI.
- Bucket S3 berikut dibuat sama Akun AWS dan Wilayah AWS sebagai perangkat inti Greengrass Anda:
  - Bucket S3 untuk menyimpan artefak yang disertakan dalam inferensi sampel dan komponen model. Tutorial ini menggunakan DOC-EXAMPLE-BUCKET1 untuk mengacu pada bucket ini.
  - Bucket S3 yang Anda kaitkan dengan armada perangkat SageMaker edge Anda. SageMaker Edge Manager memerlukan bucket S3 untuk membuat armada perangkat edge, dan menyimpan data sampel dari inferensi yang sedang berjalan di perangkat Anda. Tutorial ini menggunakan DOC-EXAMPLE-BUCKET2 untuk mengacu pada bucket ini.

  Untuk informasi selengkapnya tentang pembuatan bucket S3, lihat Memulai Amazon S3.
- Peran perangkat Greengrass yang dikonfigurasi dengan berikut ini:
  - Hubungan kepercayaan yang memungkinkan `credentials.iot.amazonaws.com` dan `sagemaker.amazonaws.com` untuk meneruskan peran, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

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

```
    }
  ]
}
```

- Kebijakan yang dikelola AmazonSageMakerEdgeDeviceFleetPolicyIAM.
- Kebijakan yang dikelola AmazonSageMakerFullAccessIAM.
- Tindakan `s3:GetObject` untuk bucket S3 yang berisi artefak komponen Anda, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Action": [
      "s3:GetObject"
    ],
    "Resource": [
      "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
    ],
    "Effect": "Allow"
  }
  ]
}
```

Siapkan perangkat inti Greengrass Anda di Edge Manager SageMaker

Armada perangkat Edge di SageMaker Edge Manager adalah kumpulan perangkat yang dikelompokkan secara logis. Untuk menggunakan SageMaker Edge Manager AWS IoT Greengrass, Anda harus membuat armada perangkat edge yang menggunakan alias AWS IoT peran yang sama dengan perangkat inti Greengrass tempat Anda menggunakan agen Edge Manager. SageMaker Kemudian, Anda harus mendaftarkan perangkat inti sebagai bagian dari armada itu.

Topik

- Buat armada perangkat edge
- Daftarkan perangkat inti Greengrass Anda

Buat armada perangkat edge

Untuk membuat armada perangkat edge (konsol)

1. Di SageMaker konsol Amazon, pilih Edge Manager, lalu pilih armada perangkat Edge.
2. Pada halaman Armada perangkat, pilih Buat armada perangkat.
3. Di bawah Properti armada perangkat, lakukan hal berikut:
   - Untuk Nama armada perangkat, masukkan nama untuk armada perangkat Anda.
   - Untuk IAM role, masukkan Amazon Resource Name (ARN) alias peran AWS IoT yang Anda tentukan saat menyiapkan perangkat inti Greengrass Anda.
   - Nonaktifkan toggle Buat alias IAM role.
4. Pilih Selanjutnya.
5. Di bawah Konfigurasi output, untuk URI bucket S3, masukkan URI bucket S3 yang ingin Anda kaitkan dengan armada perangkat tersebut.
6. Pilih Kirim.

Daftarkan perangkat inti Greengrass Anda

Untuk mendaftarkan perangkat inti Greengrass Anda sebagai perangkat edge (konsol)

1. Di SageMaker konsol Amazon, pilih Edge Manager, lalu pilih perangkat Edge.
2. Pada halaman Perangkat, pilih Daftarkan perangkat.
3. Di bawah Properti perangkat, untuk Nama armada perangkat, masukkan nama armada perangkat yang Anda buat, lalu pilih Selanjutnya.
4. Pilih Selanjutnya.
5. Di bawah Sumber perangkat, untuk nama Perangkat, masukkan nama AWS IoT benda perangkat inti Greengrass Anda.
6. Pilih Kirim.

Buat komponen sampel

Untuk membantu Anda mulai menggunakan komponen SageMaker Edge Manager, AWS sediakan skrip Python GitHub yang membuat inferensi sampel dan komponen model dan mengunggahnya ke untuk Anda. AWS Cloud Selesaikan langkah-langkah berikut pada komputer pengembangan.

Untuk membuat komponen sampel

1. Unduh repositori contoh AWS IoT Greengrass komponen GitHub ke komputer pengembangan Anda.
2. Arahkan ke unduhan folder `/machine-learning/sagemaker-edge-manager`.

   ```
   cd download-directory/machine-learning/sagemaker-edge-manager
   ```

3. Jalankan perintah berikut untuk membuat dan meng-upload komponen sampel untuk AWS Cloud.

   ```
   python3 create_components.py -r region -b DOC-EXAMPLE-BUCKET
   ```

   Ganti *wilayah* dengan Wilayah AWS tempat Anda membuat perangkat inti Greengrass, dan ganti DOC-EXAMPLE-BUCKET1 dengan nama bucket S3 untuk menyimpan artefak komponen Anda.

   > **Note**
   >
   > Secara default, skrip tersebut membuat komponen sampel baik untuk klasifikasi gambar maupun inferensi deteksi objek. Untuk membuat komponen hanya untuk jenis inferensi tertentu, tentukan argumen `-i ImageClassification | ObjectDetection`.

Inferensi sampel dan komponen model untuk digunakan dengan SageMaker Edge Manager sekarang dibuat di file Anda Akun AWS. Untuk melihat komponen sampel di Konsol AWS IoT Greengrass tersebut, pilih Komponen, dan kemudian di bawah Komponen saya, cari komponen berikut:

- `com.greengrass.SageMakerEdgeManager.ImageClassification`
- `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`
- `com.greengrass.SageMakerEdgeManager.ObjectDetection`
- `com.greengrass.SageMakerEdgeManager.ObjectDetection.Model`

Jalankan contoh inferensi klasifikasi gambar

Untuk menjalankan inferensi klasifikasi gambar menggunakan komponen sampel AWS yang disediakan dan agen SageMaker Edge Manager, Anda harus menerapkan komponen ini ke perangkat inti Anda. Menyebarkan komponen ini mengunduh model SageMaker Resnet-50 pra-terlatih yang dikompilasi NEO dan menginstal agen Edge Manager di perangkat Anda. SageMaker Agen SageMaker Edge Manager memuat model dan menerbitkan hasil inferensi pada topik tersebut. `gg/sageMakerEdgeManager/image-classification` Untuk melihat hasil inferensi ini, gunakan klien AWS IoT MQTT di AWS IoT konsol untuk berlangganan topik ini.

Topik

Berlangganan topik notifikasi

Pada langkah ini, Anda mengonfigurasi klien AWS IoT MQTT di AWS IoT konsol untuk menonton pesan MQTT yang diterbitkan oleh komponen inferensi sampel. Secara default, komponen tersebut menerbitkan hasil inferensi pada topik `gg/sageMakerEdgeManager/image-classification`. Berlangganlah topik ini sebelum Anda men-deploy komponen untuk perangkat inti Greengrass Anda untuk melihat hasil inferensi ketika komponen berjalan untuk pertama kalinya.

Untuk berlangganan topik notifikasi default

1. Di menu navigasi konsol AWS IoT tersebut, pilih Uji, klien uji MQTT.
2. Di bawah Berlangganan topik, di kotak Nama topik, masukkan **`gg/sageMakerEdgeManager/image-classification`**.
3. Pilih Langganan.

Deploy komponen sampel tersebut

Pada langkah ini, Anda mengonfigurasi dan men-deploy komponen berikut ke perangkat inti Anda:

- `aws.greengrass.SageMakerEdgeManager`
- `com.greengrass.SageMakerEdgeManager.ImageClassification`

- `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`

Untuk men-deploy komponen Anda (konsol)

1. Di menu navigasi konsol AWS IoT Greengrass tersebut, pilih Deployment, lalu pilih deployment untuk perangkat target yang ingin Anda revisi.
2. Pada halaman deployment, pilih Revisi, lalu pilih Revisi deployment.
3. Pada halaman Tentukan target, pilih Selanjutnya.
4. Pada halaman Pilih komponen, lakukan hal berikut:
   a. Di bawah Komponen saya, pilih komponen berikut:
      - `com.greengrass.SageMakerEdgeManager.ImageClassification`
      - `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`
   b. Di bawah Komponen publik, matikan toggle Tampilkan hanya komponen yang dipilih, dan kemudian pilih komponen `aws.greengrass.SageMakerEdgeManager`.
   c. Pilih Selanjutnya.
5. Pada halaman Konfigurasikan komponen, pilih komponen `aws.greengrass.SageMakerEdgeManager` dan lakukan hal berikut.
   a. Pilih Konfigurasi komponen.
   b. Di bawah Pembaruan konfigurasi, di Konfigurasi untuk menggabungkan, masukkan konfigurasi berikut.

      ```
      {
          "DeviceFleetName": "device-fleet-name",
          "BucketName": "DOC-EXAMPLE-BUCKET"
      }
      ```

      Ganti *device-fleet-name*dengan nama armada perangkat edge yang Anda buat, dan ganti *DOC-EXAMPLE-BUCKET dengan nama bucket* S3 yang terkait dengan armada perangkat Anda.
   c. Pilih Konfirmasi dan kemudian pilih Selanjutnya.
6. Pada halaman Konfigurasikan pengaturan lanjutan, simpan pengaturan konfigurasi default tersebut, dan pilih Selanjutnya.
7. Di halaman Tinjau, pilih Deploy.

Untuk men-deploy komponen Anda (AWS CLI)

1. Di komputer pengembangan Anda, buat `deployment.json` file untuk menentukan konfigurasi penerapan untuk komponen SageMaker Edge Manager Anda. File ini akan terlihat seperti contoh berikut.

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.SageMakerEdgeManager": {
      "componentVersion": "1.0.x",
      "configurationUpdate": {
        "merge": "{\"DeviceFleetName\":\"device-fleet-name\",\"BucketName\":\"DOC-
EXAMPLE-BUCKET2\"}"
      }
    },
    "com.greengrass.SageMakerEdgeManager.ImageClassification": {
      "componentVersion": "1.0.x",
      "configurationUpdate": {
      }
    },
    "com.greengrass.SageMakerEdgeManager.ImageClassification.Model": {
      "componentVersion": "1.0.x",
      "configurationUpdate": {
      }
    },
  }
}
```

   - Di kolom `targetArn`, ganti *`targetArn`* dengan Amazon Resource Name (ARN) dari grup objek atau objek yang ditargetkan untuk deployment tersebut, dalam format berikut:
     - Objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Grup objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
   - Di `merge` bidang, ganti *`device-fleet-name`*dengan nama armada perangkat tepi yang Anda buat. Kemudian, ganti*`DOC-EXAMPLE-BUCKET2`* dengan nama bucket S3 yang terkait dengan armada perangkat Anda.
   - Ganti versi komponen untuk setiap komponen dengan versi terbaru yang tersedia.
2. Jalankan perintah berikut untuk men-deploy komponen pada perangkat:

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

Deployment ini dapat memakan waktu beberapa menit hingga selesai. Pada langkah berikutnya, periksa log komponen untuk memverifikasi bahwa deployment tersebut berhasil diselesaikan dan untuk melihat hasil inferensi.

Lihat hasil inferensi

Setelah menerapkan komponen, Anda dapat melihat hasil inferensi di log komponen di perangkat inti Greengrass Anda dan di klien MQTT di konsol. AWS IoT AWS IoT Untuk berlangganan topik di mana komponen menerbitkan hasil inferensi, lihat Berlangganan topik notifikasi.

- AWS IoT Klien MQTT —Untuk melihat hasil yang diterbitkan komponen inferensi pada topik notifikasi default, selesaikan langkah-langkah berikut:

  1. Di menu navigasi konsol AWS IoT tersebut, pilih Uji, klien uji MQTT.
  2. Di bawah Langganan, pilih **gg/sageMakerEdgeManager/image-classification**.

- Log komponen—Untuk melihat hasil inferensi dalam log komponen, jalankan perintah berikut pada perangkat inti Greengrass Anda.

  ```
  sudo tail -f /greengrass/v2/logs/
  com.greengrass.SageMakerEdgeManager.ImageClassification.log
  ```

Jika Anda tidak dapat melihat hasil inferensi di log komponen atau di klien MQTT, deployment tersebut gagal atau tidak mencapai perangkat inti. Hal ini dapat terjadi jika perangkat inti Anda tidak tersambung ke internet atau tidak memiliki izin yang tepat untuk menjalankan komponen. Jalankan perintah berikut pada perangkat inti Anda untuk melihat file log perangkat lunak AWS IoT Greengrass inti. File ini mencakup log dari layanan deployment perangkat inti Greengrass.

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

Untuk informasi selengkapnya, lihat Menyelesaikan masalah inferensi machine learning.

Tutorial: Lakukan inferensi klasifikasi gambar sampel menggunakan Lite TensorFlow

Tutorial ini menunjukkan cara menggunakan komponen inferensi klasifikasi gambar TensorFlow Lite untuk melakukan inferensi klasifikasi gambar sampel pada perangkat inti Greengrass. Komponen ini mencakup dependensi komponen berikut:

- TensorFlow Komponen penyimpanan model klasifikasi gambar Lite
- TensorFlow Komponen runtime Lite

Saat Anda menerapkan komponen ini, komponen ini mengunduh model MobileNet v1 yang telah dilatih sebelumnya dan menginstal runtime TensorFlow Lite dan dependensinya. Komponen ini menerbitkan hasil inferensi pada topik `ml/tflite/image-classification`. Untuk melihat hasil inferensi ini, gunakan klien MQTT AWS IoT di konsol AWS IoT untuk berlangganan topik ini.

Dalam tutorial ini Anda akan men-deploy komponen inferensi sampel untuk melakukan klasifikasi gambar pada gambar sampel yang disediakan oleh AWS IoT Greengrass. Setelah menyelesaikan tutorial ini, Anda dapat menyelesaikan Tutorial: Lakukan inferensi klasifikasi gambar sampel pada gambar dari kamera menggunakan TensorFlow Lite, yang menunjukkan kepada Anda bagaimana memodifikasi komponen inferensi sampel untuk melakukan klasifikasi gambar pada gambar dari kamera secara lokal pada perangkat inti Greengrass.

Untuk informasi selengkapnya tentang pembelajaran mesin di perangkat Greengrass, lihat. Lakukan inferensi machine learning

Topik

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut:

- Perangkat inti Greengrass Linux. Jika Anda tidak memilikinya, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2. Perangkat inti harus memenuhi persyaratan berikut:
  - Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat tersebut.
  - Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

    ```
    sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
     libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
    ```
  - Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
    - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

      ```
      pip3 install --upgrade numpy
      ```
    - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

      Untuk mengaktifkan tumpukan kamera lama

      1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

         ```
         sudo raspi-config
         ```
      2. Pilih Opsi Antarmuka.
      3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.
      4. Reboot Raspberry Pi.

Langkah 1: Berlangganlah topik notifikasi default

Pada langkah ini, Anda mengonfigurasi klien AWS IoT MQTT di AWS IoT konsol untuk menonton pesan MQTT yang diterbitkan oleh komponen klasifikasi gambar Lite. TensorFlow Secara default, komponen tersebut menerbitkan hasil inferensi pada topik `ml/tflite/image-classification`.

Berlangganan topik ini sebelum Anda men-deploy komponen untuk perangkat inti Greengrass Anda untuk melihat hasil inferensi ketika komponen berjalan untuk pertama kalinya.

Untuk berlangganan topik notifikasi default

1. Di menu navigasi konsol AWS IoT tersebut, pilih Uji, klien uji MQTT.
2. Di bawah Berlangganan topik, di kotak Nama topik, masukkan **`ml/tflite/image-classification`**.
3. Pilih Langganan.

Langkah 2: Menerapkan komponen klasifikasi gambar TensorFlow Lite

Pada langkah ini, Anda menerapkan komponen klasifikasi gambar TensorFlow Lite ke perangkat inti Anda:

Untuk menerapkan komponen klasifikasi gambar TensorFlow Lite (konsol)

1. Pada menu navigasi konsol AWS IoT Greengrass tersebut, pilih Komponen.
2. Pada halaman Komponen, pada tab Komponen publik, pilih `aws.greengrass.TensorFlowLiteImageClassification`.
3. Pada halaman aws.greengrass.TensorFlowLiteImageClassification pilih Deploy.
4. Dari Tambahkan ke deployment, pilih salah satu langkah berikut ini:
   a. Untuk menggabungkan komponen ini ke deployment yang ada pada perangkat target Anda, pilih Tambahkan ke deployment yang ada, lalu pilih deployment yang ingin Anda revisi.
   b. Untuk membuat deployment baru di perangkat target Anda, pilih Buat deployment baru. Jika Anda memiliki deployment yang ada di perangkat, dengan memilih langkah ini Anda akan menggantikan deployment yang ada.
5. Di halaman Tentukan target, lakukan hal berikut:
   a. Di bawah informasi Deployment, masukkan atau ubah nama yang ramah untuk deployment Anda.
   b. Di bawah Target deployment, pilih target untuk deployment Anda, dan pilih Selanjutnya. Anda tidak dapat mengubah target deployment jika Anda merevisi deployment yang ada.
6. Pada halaman Pilih komponen, di bawah Komponen publik, verifikasi bahwa komponen `aws.greengrass.TensorFlowLiteImageClassification` dipilih, dan pilih Selanjutnya.

7. Pada halaman Konfigurasikan komponen, simpan pengaturan konfigurasi default, dan pilih Selanjutnya.
8. Pada halaman Konfigurasikan pengaturan lanjutan, simpan pengaturan konfigurasi default tersebut, dan pilih Selanjutnya.
9. Di halaman Tinjauan, pilih Deploy.

Untuk menerapkan komponen klasifikasi gambar TensorFlow Lite () AWS CLI

1. Buat `deployment.json` file untuk menentukan konfigurasi penerapan untuk komponen klasifikasi gambar TensorFlow Lite. File ini akan terlihat seperti berikut:

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.TensorFlowLiteImageClassification": {
      "componentVersion": 2.1.0,
      "configurationUpdate": {
      }
    }
  }
}
```

   - Di kolom `targetArn`, ganti *`targetArn`* dengan Amazon Resource Name (ARN) dari grup objek atau objek yang ditargetkan untuk deployment tersebut, dalam format berikut:
     - Objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Grup objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
   - Tutorial ini menggunakan komponen versi 2.1.0. Dalam objek `aws.greengrass.TensorFlowLiteObjectDetection` komponen, ganti *`2.1.0`* untuk menggunakan versi lain dari komponen deteksi objek TensorFlow Lite.
2. Jalankan perintah berikut untuk menerapkan komponen klasifikasi gambar TensorFlow Lite pada perangkat:

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

Deployment ini dapat memakan waktu beberapa menit hingga selesai. Pada langkah berikutnya, periksa log komponen untuk memverifikasi bahwa deployment tersebut berhasil diselesaikan dan untuk melihat hasil inferensi.

Langkah 3: Lihat hasil inferensi

Setelah Anda men-deploy komponen, Anda dapat melihat hasil inferensi dalam log komponen pada perangkat inti Greengrass Anda dan di klien MQTT AWS IoT di konsol AWS IoT tersebut. Untuk berlangganan topik di mana komponen menerbitkan hasil inferensi, lihat Langkah 1: Berlangganlah topik notifikasi default.

- Klien MQTT AWS IoT—Untuk melihat hasil yang diterbitkan oleh komponen inferensi pada topik notifikasi default, selesaikan langkah-langkah berikut:

  1. Di menu navigasi konsol AWS IoT tersebut, pilih Uji, klien uji MQTT.
  2. Di bawah Langganan, pilih **`ml/tflite/image-classification`**.

     Anda akan melihat pesan yang mirip dengan contoh berikut ini.

```
{
  "timestamp": "2021-01-01 00:00:00.000000",
  "inference-type": "image-classification",
  "inference-description": "Top 5 predictions with score 0.3 or above ",
  "inference-results": [
    {
      "Label": "cougar, puma, catamount, mountain lion, painter, panther, Felis
 concolor",
      "Score": "0.5882352941176471"
    },
    {
      "Label": "Persian cat",
      "Score": "0.5882352941176471"
    },
    {
      "Label": "tiger cat",
      "Score": "0.5882352941176471"
    },
    {
      "Label": "dalmatian, coach dog, carriage dog",
      "Score": "0.5607843137254902"
    },
    {
```

```
      "Label": "malamute, malemute, Alaskan malamute",
      "Score": "0.5450980392156862"
    }
  ]
}
```

- Log komponen—Untuk melihat hasil inferensi dalam log komponen, jalankan perintah berikut pada perangkat inti Greengrass Anda.

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

  Anda akan melihat hasil yang mirip dengan contoh berikut ini.

```
2021-01-01 00:00:00.000000 [INFO] (Copier)
 aws.greengrass.TensorFlowLiteImageClassification: stdout. Publishing results to the
 IoT core....
 {scriptName=services.aws.greengrass.TensorFlowLiteImageClassification.lifecycle.Run.script,
 serviceName=aws.greengrass.TensorFlowLiteImageClassification, currentState=RUNNING}

2021-01-01 00:00:00.000000 [INFO] (Copier)
 aws.greengrass.TensorFlowLiteImageClassification: stdout. {"timestamp":
 "2021-01-01 00:00:00.000000", "inference-type": "image-classification", "inference-
description": "Top 5 predictions with score 0.3 or above ", "inference-results":
 [{"Label": "cougar, puma, catamount, mountain lion, painter, panther, Felis
 concolor", "Score": "0.5882352941176471"}, {"Label": "Persian cat", "Score":
 "0.5882352941176471"}, {"Label": "tiger cat", "Score": "0.5882352941176471"},
 {"Label": "dalmatian, coach dog, carriage dog", "Score": "0.5607843137254902"},
 {"Label": "malamute, malemute, Alaskan malamute", "Score": "0.5450980392156862"}]}.
 {scriptName=services.aws.greengrass.TensorFlowLiteImageClassification.lifecycle.Run.script,
 serviceName=aws.greengrass.TensorFlowLiteImageClassification, currentState=RUNNING}
```

Jika Anda tidak dapat melihat hasil inferensi di log komponen atau di klien MQTT, deployment tersebut gagal atau tidak mencapai perangkat inti. Hal ini dapat terjadi jika perangkat inti Anda tidak tersambung ke internet atau tidak memiliki izin yang tepat untuk menjalankan komponen. Jalankan perintah berikut pada perangkat inti Anda untuk melihat berkas log perangkat inti AWS IoT Greengrass. File ini mencakup log dari layanan deployment perangkat inti Greengrass.

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

Untuk informasi selengkapnya, lihat Menyelesaikan masalah inferensi machine learning.

Langkah selanjutnya

Jika Anda memiliki perangkat inti Greengrass dengan antarmuka kamera yang didukung, Anda dapat menyelesaikan Tutorial: Lakukan inferensi klasifikasi gambar sampel pada gambar dari kamera menggunakan TensorFlow Lite, yang menunjukkan kepada Anda bagaimana memodifikasi komponen inferensi sampel untuk melakukan klasifikasi gambar pada gambar dari kamera.

Untuk mengeksplorasi lebih lanjut konfigurasi komponen inferensi klasifikasi gambar TensorFlow Lite sampel, coba yang berikut ini:

- Ubah parameter konfigurasi `InferenceInterval` untuk mengubah seberapa sering kode inferensi berjalan.
- Ubah parameter konfigurasi `ImageName` dan `ImageDirectory` dalam konfigurasi komponen inferensi untuk menentukan gambar kustom yang akan digunakan untuk inferensi.

Untuk informasi lebih lanjut tentang menyesuaikan konfigurasi komponen publik atau membuat komponen machine learning kustom, lihat Sesuaikan komponen machine learning Anda.

Tutorial: Lakukan inferensi klasifikasi gambar sampel pada gambar dari kamera menggunakan TensorFlow Lite

Tutorial ini menunjukkan cara menggunakan komponen inferensi klasifikasi gambar TensorFlow Lite untuk melakukan inferensi klasifikasi gambar sampel pada gambar dari kamera secara lokal pada perangkat inti Greengrass. Komponen ini mencakup dependensi komponen berikut:

- TensorFlow Komponen penyimpanan model klasifikasi gambar Lite
- TensorFlow Komponen runtime Lite

> **Note**
>
> Tutorial ini mengakses modul kamera untuk perangkat Raspberry Pi atau NVIDIA Jetson Nano, tetapi AWS IoT Greengrass mendukung perangkat lain pada platform ARMv7L, Armv8, atau x86_64. Untuk menyiapkan kamera untuk perangkat lain, lihat dokumentasi yang relevan untuk perangkat Anda.

Untuk informasi selengkapnya tentang pembelajaran mesin di perangkat Greengrass, lihat. Lakukan inferensi machine learning

Topik

Prasyarat

Untuk menyelesaikan tutorial ini, Anda harus terlebih dahulu menyelesaikan Tutorial: Lakukan inferensi klasifikasi gambar sampel menggunakan Lite TensorFlow .

Anda juga memerlukan hal berikut:

- Perangkat inti Greengrass Linux dengan antarmuka kamera. Tutorial ini mengakses modul kamera pada satu perangkat yang didukung berikut:
  - Raspberry V yang menjalankan Raspberry Pi OS (sebelumnya disebut Raspbian)
  - NVIDIA Jetson Nano

  Untuk informasi lebih lanjut tentang cara menyiapkan perangkat inti Greengrass, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2.

  Perangkat inti harus memenuhi persyaratan berikut:
  - Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat tersebut.
  - Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

    ```
    sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
     libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
    ```
  - Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:

- NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

  ```
  pip3 install --upgrade numpy
  ```

- Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

  Untuk mengaktifkan tumpukan kamera lama

  1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

     ```
     sudo raspi-config
     ```

  2. Pilih Opsi Antarmuka.
  3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.
  4. Reboot Raspberry Pi.

- Untuk perangkat Raspberry Pi atau NVIDIA Jetson Nano, Modul Kamera Raspberry Pi V2 - 8 megapiksel, 1080p. Untuk mempelajari cara mengatur kamera, lihat Menghubungkan kamera dalam dokumentasi Raspberry Pi.

Langkah 1: Konfigurasikan modul kamera pada perangkat Anda

Pada langkah ini, Anda menginstal dan mengaktifkan modul kamera untuk perangkat Anda. Jalankan perintah berikut pada perangkat.

Raspberry Pi (Armv7l)

1. Instal antarmuka `picamera` untuk modul kamera. Jalankan perintah berikut untuk menginstal modul kamera dan pustaka Python lain yang diperlukan untuk tutorial ini.

   ```
   sudo apt-get install -y python3-picamera
   ```

2. Verifikasi bahwa Picamera berhasil diinstal.

   ```
   sudo -u ggc_user bash -c 'python3 -c "import picamera"'
   ```

Jika output tidak mengandung kesalahan, validasi berhasil.

> **Note**
>
> Jika file executable Python yang diinstal pada perangkat Anda adalah `python3.7`, gunakan `python3.7` alih-alih `python3` untuk perintah di tutorial ini. Pastikan bahwa instalasi pip Anda memetakan versi `python3.7` atau `python3` untuk menghindari kesalahan dependensi.

3. Reboot perangkat.

```
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. Jalankan 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` pada direktori Anda saat ini, dan menampilkan informasi tentang kamera di terminal Raspberry Pi.

7. Jalankan perintah berikut untuk membuat symlink untuk mengaktifkan komponen inferensi untuk mengakses kamera Anda dari lingkungan virtual yang dibuat oleh komponen waktu aktif.

```
sudo ln -s /usr/lib/python3/dist-packages/picamera "MLRootPath/
greengrass_ml_tflite_venv/lib/python3.7/site-packages"
```

Nilai default untuk *ML RootPath* untuk tutorial ini adalah*/greengrass/v2*/work/`variant.TensorFlowLite/greengrass_ml`. Folder `greengrass_ml_tflite_venv` di lokasi ini dibuat ketika Anda men-deploy komponen inferensi untuk pertama kalinya di Tutorial: Lakukan inferensi klasifikasi gambar sampel menggunakan Lite TensorFlow .

Jetson Nano (Armv8)

1. Jalankan perintah berikut untuk menguji pengaturan kamera.

   ```
   gst-launch-1.0 nvarguscamerasrc num-buffers=1 ! "video/x-raw(memory:NVMM),
    width=1920, height=1080, format=NV12, framerate=30/1" ! nvjpegenc ! filesink
    location=test.jpg
   ```

   Hal ini akan menangkap dan menyimpan gambar bernama `test.jpg` pada direktori Anda saat ini.
2. (Opsional) Reboot perangkat. Jika Anda mengalami masalah saat menjalankan `gst-launch` pada langkah sebelumnya, dengan me-reboot perangkat, Anda dapat mengatasi masalah tersebut.

   ```
   sudo reboot
   ```

> **Note**
>
> Untuk perangkat Armv8 (AArch64), seperti Jetson Nano, Anda tidak perlu membuat symlink untuk mengaktifkan komponen inferensi untuk mengakses kamera dari lingkungan virtual yang dibuat oleh komponen waktu aktif.

Langkah 2: Verifikasi langganan Anda ke topik notifikasi default

DiTutorial: Lakukan inferensi klasifikasi gambar sampel menggunakan Lite TensorFlow , Anda mengonfigurasi klien AWS IoT MQTT dikonfigurasi di AWS IoT konsol untuk menonton pesan MQTT yang diterbitkan oleh komponen klasifikasi gambar TensorFlow Lite pada topik tersebut. `ml/tflite/image-classification` Di konsol AWS IoT tersebut, verifikasi bahwa langganan ini ada. Jika tidak, ikuti langkah-langkah di Langkah 1: Berlangganlah topik notifikasi default untuk berlangganan topik ini sebelum Anda men-deploy komponen ke perangkat inti Greengrass Anda.

Langkah 3: Ubah konfigurasi komponen klasifikasi gambar TensorFlow Lite dan terapkan

Pada langkah ini, Anda mengonfigurasi dan menerapkan komponen klasifikasi gambar TensorFlow Lite ke perangkat inti Anda:

Untuk mengonfigurasi dan menerapkan komponen klasifikasi gambar TensorFlow Lite (konsol)

1. Pada menu navigasi konsol AWS IoT Greengrass tersebut, pilih Komponen.
2. Pada halaman Komponen, pada tab Komponen publik, pilih `aws.greengrass.TensorFlowLiteImageClassification`.
3. Pada halaman aws.greengrass.TensorFlowLiteImageClassification pilih Deploy.
4. Dari Tambahkan ke deployment, pilih salah satu langkah berikut ini:

   a. Untuk menggabungkan komponen ini ke deployment yang ada pada perangkat target Anda, pilih Tambahkan ke deployment yang ada, lalu pilih deployment yang ingin Anda revisi.

   b. Untuk membuat deployment baru di perangkat target Anda, pilih Buat deployment baru. Jika Anda memiliki deployment yang ada di perangkat, dengan memilih langkah ini Anda akan menggantikan deployment yang ada.
5. Di halaman Tentukan target, lakukan hal berikut:

   a. Di bawah informasi Deployment, masukkan atau ubah nama yang ramah untuk deployment Anda.

   b. Di bawah Target deployment, pilih target untuk deployment Anda, dan pilih Selanjutnya. Anda tidak dapat mengubah target deployment jika Anda merevisi deployment yang ada.
6. Pada halaman Pilih komponen, di bawah Komponen publik, verifikasi bahwa komponen `aws.greengrass.TensorFlowLiteImageClassification` dipilih, dan pilih Selanjutnya.
7. Pada halaman Konfigurasikan komponen, lakukan hal berikut:

   a. Pilih komponen inferensi, dan pilih Konfigurasikan komponen.

   b. Di bawah Pembaruan konfigurasi, masukkan pembaruan konfigurasi berikut di kotak Konfigurasi yang akan digabungkan.

   ```
   {
     "InferenceInterval": "60",
     "UseCamera": "true"
   }
   ```

   Dengan pembaruan konfigurasi ini, komponen akan mengakses modul kamera pada perangkat Anda dan melakukan inferensi pada gambar yang diambil oleh kamera. Kode inferensi berjalan setiap 60 detik.

   c. Pilih Konfirmasi dan kemudian pilih Selanjutnya.

8. Pada halaman Konfigurasikan pengaturan lanjutan, simpan pengaturan konfigurasi default tersebut, dan pilih Selanjutnya.
9. Di halaman Tinjauan, pilih Deploy.

Untuk mengonfigurasi dan menerapkan komponen klasifikasi gambar TensorFlow Lite () AWS CLI

1. Buat `deployment.json` file untuk menentukan konfigurasi penerapan untuk komponen klasifikasi gambar TensorFlow Lite. File ini akan terlihat seperti berikut:

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.TensorFlowLiteImageClassification": {
      "componentVersion": 2.1.0,
      "configurationUpdate": {
        "InferenceInterval": "60",
        "UseCamera": "true"
      }
    }
  }
}
```

   - Di kolom `targetArn`, ganti *`targetArn`* dengan Amazon Resource Name (ARN) dari grup objek atau objek yang ditargetkan untuk deployment tersebut, dalam format berikut:
     - Objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Grup objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
   - Tutorial ini menggunakan komponen versi 2.1.0. Dalam objek `aws.greengrass.TensorFlowLiteImageClassification` komponen, ganti *`2.1.0`* untuk menggunakan versi lain dari komponen klasifikasi gambar TensorFlow Lite.

   Dengan pembaruan konfigurasi ini, komponen akan mengakses modul kamera pada perangkat Anda dan melakukan inferensi pada gambar yang diambil oleh kamera. Kode inferensi berjalan setiap 60 detik. Ganti nilai berikut
2. Jalankan perintah berikut untuk menerapkan komponen klasifikasi gambar TensorFlow Lite pada perangkat:

```
aws greengrassv2 create-deployment \
```

```
    --cli-input-json file://path/to/deployment.json
```

Deployment ini dapat memakan waktu beberapa menit hingga selesai. Pada langkah berikutnya, periksa log komponen untuk memverifikasi bahwa deployment tersebut berhasil diselesaikan dan untuk melihat hasil inferensi.

Langkah 4: Lihat hasil inferensi

Setelah Anda men-deploy komponen tersebut, Anda dapat melihat hasil inferensi dalam log komponen pada perangkat inti Greengrass Anda dan di klien MQTT AWS IoT di konsol AWS IoT tersebut. Untuk berlangganan topik di mana komponen menerbitkan hasil inferensi, lihat Langkah 2: Verifikasi langganan Anda ke topik notifikasi default.

- Klien MQTT AWS IoT—Untuk melihat hasil yang diterbitkan oleh komponen inferensi pada topik notifikasi default, selesaikan langkah-langkah berikut:

  1. Di menu navigasi konsol AWS IoT tersebut, pilih Uji, klien uji MQTT.
  2. Di bawah Langganan, pilih **`ml/tflite/image-classification`**.

- Log komponen—Untuk melihat hasil inferensi dalam log komponen, jalankan perintah berikut pada perangkat inti Greengrass Anda.

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

Jika Anda tidak dapat melihat hasil inferensi di log komponen atau di klien MQTT, deployment tersebut gagal atau tidak mencapai perangkat inti. Hal ini dapat terjadi jika perangkat inti Anda tidak tersambung ke internet atau tidak memiliki izin yang diperlukan untuk menjalankan komponen. Jalankan perintah berikut pada perangkat inti Anda untuk melihat berkas log perangkat inti AWS IoT Greengrass. File ini mencakup log dari layanan deployment perangkat inti Greengrass.

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

Untuk informasi selengkapnya, lihat Menyelesaikan masalah inferensi machine learning.

Langkah selanjutnya

Tutorial ini menunjukkan cara menggunakan komponen klasifikasi gambar TensorFlow Lite, dengan opsi konfigurasi khusus untuk melakukan klasifikasi gambar sampel pada gambar yang diambil oleh kamera.

Untuk informasi lebih lanjut tentang menyesuaikan konfigurasi komponen publik atau membuat komponen machine learning kustom, lihat Sesuaikan komponen machine learning Anda.

Komponen-komponen

AWS IoT Greengrasskomponen adalah modul perangkat lunak yang Anda terapkan ke perangkat inti Greengrass. Komponen dapat mewakili aplikasi, penginstal waktu aktif, pustaka, atau kode apa pun yang akan Anda jalankan pada perangkat. Anda bisa menentukan komponen yang bergantung pada komponen lainnya. Sebagai contoh, anda mungkin menentukan sebuah komponen yang menginstal Python, dan kemudian menentukan komponen tersebut sebagai dependensi dari komponen Anda yang menjalankan aplikasi Python. Ketika Anda men-deploy komponen Anda ke armada perangkat Anda, Greengrass hanya men-deploy modul perangkat lunak yang diperlukan perangkat Anda.

Topik

Komponen yang disediakan oleh AWS

AWS IoT Greengrass menyediakan dan memelihara komponen bawaan yang dapat Anda terapkan ke perangkat Anda. Komponen ini mencakup fitur (seperti manajer aliran), konektor AWS IoT Greengrass V1 (seperti CloudWatch metrik), dan alat pengembangan lokal (seperti CLI AWS IoT Greengrass). Anda dapat menerapkan komponen ini ke perangkat Anda untuk fungsionalitas mandiri mereka, atau Anda dapat menggunakannya sebagai dependensi dalam komponen Greengrass kustom Anda.

> **Note**
>
> Beberapa komponen AWS yang disediakan bergantung pada versi minor spesifik dari inti Greengrass. Karena ketergantungan ini, Anda perlu memperbarui komponen ini saat memperbarui inti Greengrass ke versi minor baru. Untuk informasi tentang versi spesifik dari inti yang masing-masing komponen bergantung padanya, lihat topik komponen yang sesuai.

Untuk informasi selengkapnya terkait cara memperbarui inti, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Ketika komponen memiliki tipe komponen generik dan Lambda, versi komponen saat ini adalah tipe generik dan versi komponen sebelumnya adalah tipe Lambda.

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Inti Greengrass	Inti dari perangkat lunak AWS IoT Greengrass Inti. Gunakan komponen ini untuk mengonfig urasi dan memperbar ui perangkat lunak pada perangkat inti Anda.	Inti	Linux, Windows	Ya
Auth perangkat klien	Memungkin kan perangkat IoT lokal, yang disebut perangkat klien, untuk terhubung ke perangkat inti.	Plugin	Linux, Windows	Ya

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
CloudWatch metrik	Menerbitkan metrik khusus ke Amazon. CloudWatch	Generik, Lambda	Linux, Windows	Ya
AWS IoT Device Defender	Memberitahu administrator tentang perubahan pada keadaan perangkat inti Greengrass untuk mengidentifikasi perilaku yang tidak biasa.	Generik, Lambda	Linux, Windows	Ya

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Spooler disk	Mengaktifkan opsi penyimpanan persisten untuk pesan yang di-spooled dari perangkat inti Greengrass ke. AWS IoT Core Komponen ini akan menyimpan pesan keluar ini pada disk.	Plugin	Linux, Windows	Ya
Manajer aplikasi Docker	Memungkinkan AWS IoT Greengrass untuk mengunduh gambar Docker dari Docker Hub dan Amazon Elastic Container Registry (Amazon ECR).	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Konektor tepi untuk Kinesis Video Streams	Membaca umpan video dari kamera lokal, menerbitkan aliran ke Kinesis Video Streams, dan menampilkan aliran di dasbor Grafana. AWS IoT TwinMaker	Generik	Linux	Tidak
Greengrass CLI	Menyediakan antarmuka baris perintah yang dapat Anda gunakan untuk membuat deployment lokal dan berinteraksi dengan perangkat inti Greengrass dan komponennya.	Plugin	Linux, Windows	Ya

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Detektor IP	Melaporkan informasi konektivitas broker MQTT ke AWS IoT Greengrass, sehingga perangkat klien dapat menemukan cara terhubung.	Plugin	Linux, Windows	Ya
Firehose	Menerbitkan data melalui aliran pengiriman Amazon Data Firehose ke tujuan di. AWS Cloud	Lambda	Linux	Tidak
Peluncur Lambda	Menangani proses dan konfigurasi lingkungan untuk fungsi Lambda.	Generik	Linux	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Manajer Lambda	Menangani komunikasi antar proses dan penskalaan untuk fungsi Lambda.	Plugin	Linux	Tidak
Runtime Lambda	Memberikan artefak untuk setiap waktu aktif Lambda.	Generik	Linux	Tidak
Router langganan warisan	Mengelola langganan untuk fungsi Lambda yang berjalan AWS IoT Greengrass di V1.	Generik	Linux	Tidak
Konsol debug lokal	Menyediakan konsol lokal yang dapat Anda gunakan untuk debug dan mengelola perangkat inti Greengrass dan komponennya.	Plugin	Linux, Windows	Ya

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Manajer log	Mengumpulkan dan mengunggah log pada perangkat inti Greengrass.	Plugin	Linux, Windows	Ya
Komponen machine learning	Menyediakan model machine learning dan contoh kode inferensi yang dapat Anda gunakan untuk melakukan inferensi machine learning pada perangkat inti Greengrass.	Lihat Komponen machine learning.		
Adaptor protokol Modbus-RTU	Mengumpulkan informasi dari perangkat Modbus RTU lokal.	Lambda	Linux	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Pemancar telemetri nukleus	Menerbitkan data telemetri kesehatan sistem yang dikumpulkan dari inti ke topik lokal atau ke topik MQTT. AWS IoT Core	Plugin	Linux, Windows	Ya
Jembatan MQTT	Relay pesan MQTT antara perangkat klien, penerbitan/berlangganan lokal AWS IoT Greengrass , dan. AWS IoT Core	Plugin	Linux, Windows	Ya
MQTT 3.1.1 broker (Moquette)	Menjalankan broker MQTT 3.1.1 yang menangani pesan antara perangkat klien dan perangkat inti.	Plugin	Linux, Windows	Ya

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Pialang MQTT 5 (EMQX)	Menjalankan broker MQTT 5 yang menangani pesan antara perangkat klien dan perangkat inti.	Generik	Linux, Windows	Tidak
Penyedia PKCS #11	Mengaktifkan komponen Greengrass untuk mengakses kunci pribadi dan sertifikat yang Anda simpan dengan aman di modul keamanan perangkat keras (HSM).	Plugin	Linux	Ya

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Secrets manager	Menyebarkan rahasia dari AWS Secrets Manager rahasia sehingga Anda dapat menggunakan kredensil dengan aman, seperti kata sandi, dalam komponen khusus pada perangkat inti Greengrass.	Plugin	Linux, Windows	Ya

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Tunneling yang aman	Mengaktifkan koneksi tunneling AWS IoT aman yang dapat Anda gunakan untuk membuat komunikasi bidirectional dengan perangkat inti Greengrass yang berada di belakang firewall terbatas.	Generik	Linux	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Pengelola bayangan	Memungkinkan interaksi dengan bayangan pada perangkat inti. Ini mengelola penyimpanan dokumen bayangan dan juga sinkronisasi status bayangan lokal dengan layanan AWS IoT Device Shadow.	Plugin	Linux, Windows	Ya
Amazon SNS	Menerbitkan pesan ke topik Amazon SNS.	Lambda	Linux	Tidak
Manajer pengaliran	Streaming data bervolume tinggi dari sumber lokal ke AWS Cloud.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Agen Systems Manager	Kelola perangkat inti dengan AWS Systems Manager, yang memungkinkan Anda untuk menambal perangkat, menjalankan perintah, dan banyak lagi.	Generik	Linux	Tidak
Layanan pertukaran token	Memberikan AWS kredensi yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan.	Generik	Linux, Windows	Tidak
Kolektor IoT SiteWise OPC-UA	Mengumpulkan data dari server OPC-UA.	Generik	Linux, Windows	Tidak
Simulator sumber data IoT SiteWise OPC-UA	Menjalankan server OPC-UA lokal yang menghasilkan data sampel.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Penerbit IoT SiteWise	Mempublik asikan data ke AWS Cloud.	Generik	Linux, Windows	Tidak
Prosesor IoT SiteWise	Memproses data pada perangkat inti Greengrass.	Generik	Linux, Windows	Tidak

Inti Greengrass

Komponen inti Greengrass `aws.greengrass.Nucleus` () adalah komponen wajib dan persyaratan minimum untuk menjalankan perangkat lunak Core AWS IoT Greengrass pada perangkat. Anda dapat mengonfigurasi komponen ini untuk menyesuaikan dan memperbarui perangkat lunak AWS IoT Greengrass Core Anda dari jarak jauh. Terapkan komponen ini untuk mengonfigurasi pengaturan seperti proxy, peran perangkat, dan konfigurasi AWS IoT benda pada perangkat inti Anda.

Important

Saat versi komponen nukleus berubah, atau saat Anda mengubah parameter konfigurasi tertentu, perangkat lunak AWS IoT Greengrass Inti—yang menyertakan inti dan semua komponen lain di perangkat Anda—akan dimulai ulang untuk menerapkan perubahan.
Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena itu, versi patch baru dari komponen publik AWS yang disediakan mungkin secara otomatis diterapkan ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup sesuatu, atau Anda memperbarui penerapan yang menargetkan perangkat tersebut. Beberapa pembaruan otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk

perangkat lunak AWS IoT Greengrass Core, lihatPerbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Topik

- Versi
- Sistem operasi
- Persyaratan
- Dependensi
- Download dan Instalasi
- Konfigurasi
- Berkas log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.12.x
- 2.11.x
- 2.10.x
- 2.9.x
- 2.8.x
- 2.7.x
- 2.6.x
- 2.5.x
- 2.4.x
- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Untuk informasi selengkapnya, lihat Platform yang didukung.

Persyaratan

Perangkat harus memenuhi persyaratan tertentu untuk menginstal dan menjalankan inti Greengrass dan perangkat lunak Core. AWS IoT Greengrass Untuk informasi selengkapnya, lihat Persyaratan perangkat.

Komponen inti Greengrass didukung untuk berjalan di VPC. Untuk menerapkan komponen ini di VPC, berikut ini diperlukan.

- Komponen inti Greengrass harus memiliki konektivitas AWS IoT data ke, AWS IoT Credentials, dan Amazon S3.

Dependensi

Inti Greengrass tidak termasuk dependensi komponen apapun. Namun, beberapa komponen yang disediakan oleh AWS mencakup nukleus sebagai dependensi. Untuk informasi selengkapnya, lihat Komponen yang disediakan oleh AWS.

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Download dan Instalasi

Anda dapat men-download installer yang mengatur komponen inti Greengrass pada perangkat Anda. Installer ini mengatur perangkat Anda sebagai perangkat inti Greengrass. Ada dua jenis instalasi yang dapat Anda lakukan: instalasi cepat yang menciptakan AWS sumber daya yang diperlukan untuk Anda, atau instalasi manual di mana Anda membuat AWS sumber daya sendiri. Untuk informasi selengkapnya, lihat Instal perangkat lunak inti AWS IoT Greengrass.

Anda juga dapat mengikuti tutorial untuk menginstal inti Greengrass dan mengeksplorasi pengembangan komponen Greengrass. Untuk informasi selengkapnya, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen. Beberapa parameter mengharuskan perangkat lunak AWS IoT Greengrass Core restart untuk diterapkan. Untuk informasi selengkapnya tentang mengapa dan bagaimana komponen ini, lihat Konfigurasikan perangkat lunak AWS IoT Greengrass Inti.

`iotRoleAlias`

Alias AWS IoT peran yang menunjuk ke peran IAM pertukaran token. Penyedia AWS IoT kredensi mengasumsikan peran ini untuk memungkinkan perangkat inti Greengrass berinteraksi dengan layanan. AWS Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Ketika Anda menjalankan perangkat lunak AWS IoT Greengrass Core dengan `--provision true` opsi, perangkat lunak menyediakan alias peran dan menetapkan nilainya dalam komponen inti.

`interpolateComponentConfiguration`

(Opsional) Anda dapat mengaktifkan inti Greengrass untuk menginterpolasi variabel resep komponen dalam konfigurasi komponen dan menggabungkan pembaruan konfigurasi. Kami menyarankan Anda mengatur opsi ini `true` agar perangkat inti dapat menjalankan komponen Greengrass yang menggunakan variabel resep dalam konfigurasinya.

Fitur ini tersedia untuk v2.6.0 dan yang lebih baru dari komponen ini.

Default: `false`

`networkProxy`

(Opsional) Proksi jaringan yang digunakan untuk semua koneksi. Untuk informasi selengkapnya, lihat Hubungkan pada port 443 atau melalui proksi jaringan.

> **Important**
>
> Saat Anda menerapkan perubahan pada parameter konfigurasi ini, perangkat lunak AWS IoT Greengrass Core akan dimulai ulang agar perubahan diterapkan.

Objek ini berisi informasi berikut:

`noProxyAddresses`

(Opsional) Daftar alamat IP atau nama host yang dipisahkan koma yang dikecualikan dari proxy.

`proxy`

Proksi yang akan dihubungkan. Objek ini berisi informasi berikut:

`url`

URL server proksi dalam format `scheme://userinfo@host:port`.

- `scheme` — Skema, yang harus berupa `http` atau `https`.

  > **Important**
  >
  > Perangkat inti Greengrass harus menjalankan Greengrass nucleus v2.5.0 atau yang lebih baru untuk menggunakan proxy HTTPS.
  > Jika Anda mengonfigurasi proxy HTTPS, Anda harus menambahkan sertifikat CA server proxy ke sertifikat CA root Amazon perangkat inti. Untuk informasi selengkapnya, lihat Aktifkan perangkat inti untuk mempercayai proxy HTTPS.

- `userinfo` - (Opsional) Nama pengguna dan informasi kata sandi. Jika Anda menentukan informasi ini di`url`, perangkat inti Greengrass mengabaikan bidang dan. `username` `password`
- `host` - Nama host atau alamat IP server proksi.
- `port` — (Opsional) Nomor port. Jika Anda tidak menentukan port, maka perangkat inti Greengrass akan menggunakan nilai default berikut:
  - `http` – 80
  - `https` – 443

`username`

(Opsional) Nama pengguna yang mengautentikasi server proxy.

`password`

(Opsional) Kata sandi yang mengautentikasi server proxy.

`mqtt`

(Opsional) Konfigurasi MQTT untuk perangkat inti Greengrass. Untuk informasi selengkapnya, lihat Hubungkan pada port 443 atau melalui proksi jaringan.

> **Important**
>
> Saat Anda menerapkan perubahan pada parameter konfigurasi ini, perangkat lunak AWS IoT Greengrass Core akan dimulai ulang agar perubahan diterapkan.

Objek ini berisi informasi berikut:

`port`

(Opsional) Port yang akan digunakan untuk koneksi MQTT.

Default: `8883`

`keepAliveTimeoutMs`

(Opsional) Jumlah waktu dalam milidetik antara setiap pesan `PING` yang dikirim klien untuk menjaga sambungan MQTT hidup. Nilai ini harus lebih besar dari`pingTimeoutMs`.

Default: `60000` (60 detik)

`pingTimeoutMs`

(Opsional) Jumlah waktu dalam milidetik yang klien tunggu untuk menerima pesan `PINGACK` dari server. Jika masa tunggu melebihi batas waktu, perangkat inti menutup dan membuka kembali sambungan MQTT. Nilai ini harus kurang dari`keepAliveTimeoutMs`.

Default: `30000` (30 detik)

`operationTimeoutMs`

(Opsional) Jumlah waktu dalam milidetik yang klien menunggu operasi MQTT (seperti `CONNECT` atau) untuk diselesaikan. `PUBLISH` Opsi ini tidak berlaku untuk MQTT `PING` atau pesan tetap hidup.

Default: `30000` (30 detik)

`maxInFlightPublishes`

(Opsional) Jumlah maksimum pesan MQTT QoS 1 yang tidak diakui yang dapat terbang pada waktu yang sama.

Fitur ini tersedia untuk v2.1.0 dan versi kemudian dari komponen ini.

Default: 5

Kisaran valid: Nilai maksimum 100

`maxMessageSizeInBytes`

(Opsional) Ukuran maksimum pesan MQTT. Jika pesan melebihi ukuran ini, inti Greengrass menolak pesan dengan kesalahan.

Fitur ini tersedia untuk v2.1.0 dan versi kemudian dari komponen ini.

Default: `131072` (128 KB)

Kisaran valid: Nilai maksimum `2621440` (2,5 MB)

`maxPublishRetry`

(Opsional) Jumlah waktu maksimum untuk mencoba kembali pesan yang gagal untuk dipublikasikan. Anda dapat menentukan `-1` untuk mencoba lagi dalam waktu tak terbatas.

Fitur ini tersedia untuk v2.1.0 dan versi kemudian dari komponen ini.

Default: `100`

`spooler`

(Opsional) Konfigurasi spooler MQTT untuk perangkat inti Greengrass. Objek ini berisi informasi berikut:

`storageType`

Jenis penyimpanan untuk menyimpan pesan. Jika `storageType` diatur ke`Disk`, `pluginName` dapat dikonfigurasi. Anda dapat menentukan `Memory` atau `Disk`.

Fitur ini tersedia untuk v2.11.0 dan yang lebih baru dari komponen inti Greengrass.

> **Important**
>
> Jika spooler MQTT `storageType` disetel ke `Disk` dan Anda ingin menurunkan versi inti Greengrass dari versi 2.11.x ke versi sebelumnya, Anda harus mengubah konfigurasi kembali ke. `Memory` Satu-satunya konfigurasi untuk `storageType` itu didukung di Greengrass nucleus versi 2.10.x dan sebelumnya adalah. `Memory` Tidak mengikuti panduan ini dapat mengakibatkan spooler rusak. Ini akan

menyebabkan perangkat inti Greengrass Anda tidak dapat mengirim pesan MQTT ke. AWS Cloud

Default: `Memory`

`pluginName`

(Opsional) Nama komponen plugin. Komponen ini hanya akan digunakan jika `storageType` diatur ke`Disk`. Opsi ini default ke `aws.greengrass.DiskSpooler` dan akan menggunakan GreenGrass yang disediakan. Spooler disk

Fitur ini tersedia untuk v2.11.0 dan yang lebih baru dari komponen inti Greengrass.

Default: `"aws.greengrass.DiskSpooler"`

`maxSizeInBytes`

(Opsional) Ukuran maksimum cache di mana perangkat inti menyimpan pesan MQTT yang belum diproses dalam memori. Jika cache penuh, pesan baru ditolak.

Default: `2621440` (2,5 MB)

`keepQos0WhenOffline`

(Opsional) Anda dapat mengirim pesan MQTT QoS 0 yang diterima oleh perangkat inti ketika ia offline. Jika Anda menetapkan opsi ini ke `true`, perangkat inti akan mengirim pesan QoS 0 yang tidak dapat dikirimnya saat sedang offline. Jika Anda menetapkan opsi ini ke `false`, perangkat inti akan membuang pesan ini. Perangkat inti selalu mengirim pesan QoS 1 pesan kecuali spool penuh.

Default: `false`

`version`

(Opsional) Versi MQTT. Anda dapat menentukan `mqtt3` atau `mqtt5`.

Fitur ini tersedia untuk v2.10.0 dan yang lebih baru dari komponen inti Greengrass.

Default: `mqtt5`

`receiveMaximum`

(Opsional) Jumlah maksimum paket QoS1 yang tidak diakui yang dapat dikirim oleh broker.

Fitur ini tersedia untuk v2.10.0 dan yang lebih baru dari komponen inti Greengrass.

Default: `100`

`sessionExpirySeconds`

(Opsional) Jumlah waktu dalam hitungan detik yang dapat Anda minta agar sesi berlangsung dari IoT Core. Defaultnya adalah waktu maksimum yang didukung oleh AWS IoT Core.

Fitur ini tersedia untuk v2.10.0 dan yang lebih baru dari komponen inti Greengrass.

Default: `604800 (7 days)`

`minimumReconnectDelaySeconds`

(Opsional) Opsi untuk perilaku rekoneksi. Jumlah minimum waktu dalam hitungan detik untuk MQTT untuk menyambung kembali.

Fitur ini tersedia untuk v2.10.0 dan yang lebih baru dari komponen inti Greengrass.

Default: `1`

`maximumReconnectDelaySeconds`

(Opsional) Opsi untuk perilaku rekoneksi. Jumlah maksimum waktu dalam hitungan detik untuk MQTT untuk menyambung kembali.

Fitur ini tersedia untuk v2.10.0 dan yang lebih baru dari komponen inti Greengrass.

Default: `120`

`minimumConnectedTimeBeforeRetryResetSeconds`

(Opsional) Opsi untuk perilaku rekoneksi. Jumlah waktu dalam hitungan detik koneksi harus aktif sebelum penundaan coba lagi diatur ulang kembali ke minimum.

Fitur ini tersedia untuk v2.10.0 dan yang lebih baru dari komponen inti Greengrass.

Default: `30`

`jvmOptions`

(Opsional) Opsi JVM yang akan digunakan untuk menjalankan perangkat lunak AWS IoT Greengrass Core. Untuk informasi tentang opsi JVM yang direkomendasikan untuk menjalankan perangkat lunak AWS IoT Greengrass Core, lihat. Kontrol alokasi memori dengan opsi JVM

> **Important**
>
> Saat Anda menerapkan perubahan pada parameter konfigurasi ini, perangkat lunak AWS IoT Greengrass Core akan dimulai ulang agar perubahan diterapkan.

`iotDataEndpoint`

Titik akhir AWS IoT data untuk Anda Akun AWS.

Ketika Anda menjalankan perangkat lunak AWS IoT Greengrass Core dengan `--provision true` opsi, perangkat lunak mendapatkan data dan titik akhir kredensial Anda dari AWS IoT dan mengaturnya dalam komponen nukleus.

`iotCredEndpoint`

Titik akhir AWS IoT kredensial untuk Anda. Akun AWS

Ketika Anda menjalankan perangkat lunak AWS IoT Greengrass Core dengan `--provision true` opsi, perangkat lunak mendapatkan data dan titik akhir kredensial Anda dari AWS IoT dan mengaturnya dalam komponen nukleus.

`greengrassDataPlaneEndpoint`

Fitur ini tersedia di v2.7.0 dan yang lebih baru dari komponen ini.

Untuk informasi selengkapnya, lihat Menggunakan sertifikat perangkat yang ditandatangani oleh CA pribadi.

`greengrassDataPlanePort`

Fitur ini tersedia di v2.0.4 dan yang lebih baru dari komponen ini.

(Opsional) Port yang akan digunakan untuk koneksi bidang data. Untuk informasi selengkapnya, lihat Hubungkan pada port 443 atau melalui proksi jaringan.

> **Important**
>
> Anda harus menentukan port di mana perangkat dapat membuat koneksi keluar. Jika Anda menentukan port yang diblokir, perangkat tidak akan dapat terhubung AWS IoT Greengrass untuk menerima penerapan.

Pilih dari salah satu pilihan berikut:

- `443`
- `8443`

Default: `8443`

`awsRegion`

Wilayah AWS Untuk digunakan.

`runWithDefault`

Pengguna sistem yang digunakan untuk menjalankan komponen.

> **Important**
>
> Saat Anda menerapkan perubahan pada parameter konfigurasi ini, perangkat lunak AWS IoT Greengrass Core akan dimulai ulang agar perubahan diterapkan.

Objek ini berisi informasi berikut:

`posixUser`

Nama atau ID pengguna sistem dan, secara opsional, grup sistem yang digunakan perangkat inti untuk menjalankan komponen generik dan Lambda. Tentukan pengguna dan grup yang dipisahkan dengan titik dua (`:`) dalam format berikut: `user:group`. Grup ini opsional. Jika Anda tidak menentukan grup, perangkat lunak AWS IoT Greengrass Core menggunakan grup utama untuk pengguna. Misalnya, Anda dapat menentukan `ggc_user` atau `ggc_user:ggc_group`. Untuk informasi selengkapnya, lihat Konfigurasikan pengguna yang menjalankan komponen.

Saat Anda menjalankan penginstal perangkat lunak AWS IoT Greengrass Core dengan `--component-default-user` *`ggc_user`*`:`*`ggc_group`* opsi, perangkat lunak menetapkan parameter ini di komponen inti.

`windowsUser`

Fitur ini tersedia di v2.5.0 dan yang lebih baru dari komponen ini.

Nama pengguna Windows yang digunakan untuk menjalankan komponen ini pada perangkat inti Windows. Pengguna harus ada di setiap perangkat inti Windows, dan nama serta kata

sandinya harus disimpan dalam instance Credentials Manager LocalSystem akun. Untuk informasi selengkapnya, lihat Konfigurasikan pengguna yang menjalankan komponen.

Saat Anda menjalankan penginstal perangkat lunak AWS IoT Greengrass Core dengan `--component-default-user` *`ggc_user`* opsi, perangkat lunak menetapkan parameter ini di komponen inti.

`systemResourceLimits`

Fitur ini tersedia di v2.4.0 dan yang lebih baru dari komponen ini. AWS IoT Greengrass saat ini tidak mendukung fitur ini di perangkat inti Windows.

Batas sumber daya sistem untuk diterapkan pada proses komponen Lambda generik dan non-kontainer secara default. Anda dapat mengganti batas sumber daya sistem untuk masing-masing komponen saat membuat penerapan. Untuk informasi selengkapnya, lihat Konfigurasikan batas sumber daya sistem untuk komponen.

Objek ini berisi informasi berikut:

`cpus`

Jumlah maksimum waktu CPU yang dapat digunakan oleh setiap proses komponen pada perangkat inti. Total waktu CPU perangkat inti setara dengan jumlah inti CPU perangkat. Misalnya, pada perangkat inti dengan 4 core CPU, Anda dapat mengatur nilai ini `2` untuk membatasi proses setiap komponen hingga 50 persen penggunaan setiap inti CPU. Pada perangkat dengan 1 inti CPU, Anda dapat mengatur nilai ini `0.25` untuk membatasi proses setiap komponen hingga 25 persen penggunaan CPU. Jika Anda menetapkan nilai ini ke angka yang lebih besar dari jumlah inti CPU, perangkat lunak AWS IoT Greengrass Core tidak membatasi penggunaan CPU komponen.

`memory`

Jumlah maksimum RAM (dalam kilobyte) yang dapat digunakan oleh setiap proses komponen pada perangkat inti.

`s3EndpointType`

(Opsional) Jenis titik akhir S3. Parameter ini hanya akan berlaku untuk Wilayah AS Timur (Virginia N.) (`us-east-1`). Pengaturan parameter ini dari Wilayah lain akan diabaikan. Pilih dari salah satu pilihan berikut:

- `REGIONAL`— Klien S3 dan URL presigned menggunakan endpoint regional.

- `GLOBAL`— Klien S3 dan URL presigned menggunakan endpoint lama.

Default: `GLOBAL`

`logging`

(Opsional) Konfigurasi pencatatan untuk perangkat inti. Untuk informasi selengkapnya tentang cara mengonfigurasi dan menggunakan log Greengrass, lihat. Memantau AWS IoT Greengrass log

Objek ini berisi informasi berikut:

`level`

(Opsional) Tingkat minimum pesan log untuk output.

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`format`

(Opsional) Format data log. Pilih dari salah satu pilihan berikut:

- `TEXT`— Pilih opsi ini jika Anda ingin melihat log dalam bentuk teks.
- `JSON`— Pilih opsi ini jika Anda ingin melihat log dengan perintah log CLI Greengrass atau berinteraksi dengan log secara terprogram.

Default: `TEXT`

`outputType`

(Opsional) Jenis output untuk log. Pilih dari salah satu pilihan berikut:

- `FILE`— Perangkat lunak AWS IoT Greengrass Core mengeluarkan log ke file di direktori yang Anda tentukan. `outputDirectory`
- `CONSOLE`— Perangkat lunak AWS IoT Greengrass Core mencetak log `kestdout`. Pilih opsi ini untuk melihat log begitu perangkat inti mencetaknya.

Default: `FILE`

`fileSizeKB`

(Opsional) Ukuran maksimum setiap file log (dalam kilobyte). Setelah file log melebihi ukuran file maksimum ini, perangkat lunak AWS IoT Greengrass Core membuat file log baru.

Parameter ini hanya berlaku bila Anda menentukan `FILE` untuk `outputType`.

Default: `1024`

`totalLogsSizeKB`

(Opsional) Ukuran total maksimum file log (dalam kilobyte) untuk setiap komponen, termasuk inti Greengrass. File log Greengrass nucleus juga menyertakan log dari komponen plugin. Setelah ukuran total file log komponen melebihi ukuran maksimum ini, perangkat lunak AWS IoT Greengrass Core menghapus file log tertua komponen tersebut.

Parameter ini setara dengan parameter batas ruang disk komponen pengelola log (`diskSpaceLimit`), yang dapat Anda tentukan untuk inti Greengrass (sistem) dan setiap komponen. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log total maksimum untuk inti Greengrass dan setiap komponen.

Parameter ini hanya berlaku bila Anda menentukan `FILE` untuk `outputType`.

Default: `10240`

`outputDirectory`

(Opsional) Direktori output untuk file log.

Parameter ini hanya berlaku bila Anda menentukan `FILE` untuk `outputType`.

Default:*`/greengrass/v2`*`/logs`, di mana *`/greengrass/v2`* folder AWS IoT Greengrass root.

`fleetstatus`

Parameter ini tersedia pada v2.1.0 dan versi yang lebih baru dari komponen ini.

(Opsional) Konfigurasi status armada untuk perangkat inti.

Objek ini berisi informasi berikut:

`periodicStatusPublishIntervalSeconds`

(Opsional) Jumlah waktu (dalam detik) yang di antaranya perangkat inti menerbitkan status perangkat ke AWS Cloud.

Minimal: `86400` (24 jam)

Default: `86400` (24 jam)

`telemetry`

(Opsional) Konfigurasi telemetri kesehatan sistem untuk perangkat inti. Untuk informasi lebih lanjut tentang metrik telemetri dan cara memperlakukan data telemetri, lihat Kumpulkan data telemetri kondisi sistem dari perangkat inti AWS IoT Greengrass.

Objek ini berisi informasi berikut:

`enabled`

(Opsional) Anda dapat mengaktifkan atau menonaktifkan telemetri.

Default: `true`

`periodicAggregateMetricsIntervalSeconds`

(Opsional) Interval (dalam detik) di mana perangkat inti menggabungkan metrik.

Jika Anda menetapkan nilai ini lebih rendah dari nilai minimum yang didukung, nukleus akan menggunakan nilai default sebagai gantinya.

Minimal: `3600`

Default: `3600`

`periodicPublishMetricsIntervalSeconds`

(Opsional) Jumlah waktu (dalam detik) yang di antaranya perangkat inti menerbitkan metrik telemetri ke AWS Cloud.

Jika Anda menetapkan nilai ini lebih rendah dari nilai minimum yang didukung, nukleus akan menggunakan nilai default sebagai gantinya.

Minimal: `86400`

Default: `86400`

`deploymentPollingFrequencySeconds`

(Opsional) Periode dalam detik untuk mengumpulkan notifikasi deployment.

Default: `15`

`componentStoreMaxSizeBytes`

(Opsional) Ukuran maksimum pada disk dari penyimpanan komponen, yang terdiri dari resep komponen dan artefak.

Default: `10000000000` (10 GB)

`platformOverride`

(Opsional) Kamus atribut yang mengidentifikasi platform perangkat inti. Gunakan ini untuk menentukan atribut platform kustom yang dapat digunakan resep komponen untuk mengidentifikasi siklus hidup dan artefak yang benar untuk komponen. Misalnya, Anda mungkin menentukan atribut kemampuan perangkat keras untuk men-deploy hanya serangkaian artefak minimal yang akan dijalankan oleh komponen. Untuk informasi lebih lanjut, lihat parameter platform manifes dalam resep komponen.

Anda juga dapat menggunakan parameter ini untuk menimpa atribut platform `os` dan `architecture` dari perangkat inti.

`httpClient`

Parameter ini tersedia di v2.5.0 dan yang lebih baru dari komponen ini.

(Opsional) Konfigurasi klien HTTP untuk perangkat inti. Opsi konfigurasi ini berlaku untuk semua permintaan HTTP yang dibuat oleh komponen ini. Jika perangkat inti berjalan pada jaringan yang lebih lambat, Anda dapat meningkatkan durasi waktu tunggu ini untuk mencegah permintaan HTTP dari waktu habis.

Objek ini berisi informasi berikut:

`connectionTimeoutMs`

(Opsional) Jumlah waktu (dalam milidetik) untuk menunggu koneksi terbuka sebelum waktu permintaan koneksi habis.

Default: `2000` (2 detik)

`socketTimeoutMs`

(Opsional) Jumlah waktu (dalam milidetik) untuk menunggu data ditransfer melalui koneksi terbuka sebelum waktu koneksi habis.

Default: `30000` (30 detik)

Example Contoh: Pembaruan gabungan konfigurasi

```
{
  "iotRoleAlias": "GreengrassCoreTokenExchangeRoleAlias",
  "networkProxy": {
    "noProxyAddresses": "http://192.168.0.1,www.example.com",
    "proxy": {
      "url": "http://my-proxy-server:1100",
      "username": "Mary_Major",
      "password": "pass@word1357"
    }
  },
  "mqtt": {
    "port": 443
  },
  "greengrassDataPlanePort": 443,
  "jvmOptions": "-Xmx64m",
  "runWithDefault": {
    "posixUser": "ggc_user:ggc_group"
  }
}
```

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.12.6	Perbaikan bug dan peningkatan • Memperbaiki masalah yang menyebabkan crash saat startup pada prosesor ARMv8 tertentu, termasuk Jetson Nano.
2.12.5	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana rollback penerapan terkadang macet saat memutar kembali komponen yang sebelumnya rusak dengan dependensi keras. • Memperbaiki masalah di mana nukleus tidak mempublikasikan pembaruan status setelah penyediaan armada. • Menambahkan percobaan ulang untuk `GetDeploymentConfiguration` API setelah mendapatkan 404 kesalahan.
2.12.4	Perbaikan bug dan peningkatan • Memperbaiki masalah saat nukleus memasuki kondisi kebuntuan selama startup di beberapa perangkat Linux.

Versi	Perubahan
2.12.3	**Warning** Versi ini tidak lagi tersedia. Perbaikan dalam versi ini tersedia di versi yang lebih baru dari komponen ini. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana nukleus tidak melaporkan status komponen yang benar setelah nukleus diluncurkan kembali dan selama pemulihan komponen. • Perbaikan bug umum dan perbaikan.
2.12.2	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana log lama tidak dibersihkan dengan benar. • Perbaikan bug umum dan perbaikan.
2.12.1	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana nukleus dapat menduplikasi langganan MQTT ke topik penerapan yang mengarah ke pencatatan tambahan dan penerbitan MQTT.
2.12.0	Fitur baru • Memungkinkan Anda menjalankan langkah-langkah siklus hidup bootstrap sebagai bagian dari penerapan rollback.
2.11.3	Perbaikan bug dan peningkatan • Memperbaiki masalah di nukleus di mana ia mungkin memulai komponen dengan tidak benar ketika dependensinya gagal. Fitur baru • Menambahkan tipe endpoint s3 yang dapat dikonfigurasi.

Versi	Perubahan
2.11.2	Perbaikan bug dan peningkatan • Memperbaiki masalah di klien nukleus MQTT 5 yang mungkin muncul offline saat sejumlah besar (> 50) langganan sedang digunakan. • Menambahkan percobaan lagi untuk kegagalan TCP docker dial.
2.11.1	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana nukleus tidak dimulai jika tugas bootstrap gagal dan file metadata penerapan rusak. • Memperbaiki masalah saat komponen Lambda sesuai permintaan tidak dilaporkan dalam pembaruan status penerapan. • Menambahkan dukungan untuk ID kebijakan otorisasi duplikat.
2.11.0	Fitur baru • Memungkinkan Anda membatalkan penerapan lokal. • Memungkinkan Anda mengonfigurasi kebijakan penanganan kegagalan untuk penerapan lokal. • Menambahkan dukungan untuk plugin spooler disk.
2.10.3	Perbaikan bug dan peningkatan • Memperbaiki masalah saat Greengrass tidak berlangganan notifikasi penerapan saat menggunakan penyedia PKCS #11.
2.10.2	Perbaikan bug dan peningkatan • Memungkinkan penguraian siklus hidup komponen yang tidak peka huruf besar/kecil. • Memperbaiki masalah di mana variabel PATH lingkungan tidak dibuat ulang dengan benar. • Memperbaiki pengkodean URI proxy untuk komponen termasuk manajer aliran untuk nama pengguna dengan karakter khusus.

Versi	Perubahan
2.10.1	Perbaikan bug dan peningkatan • Memperbaiki masalah yang dapat menyebabkan crash saat startup pada prosesor ARMv8 tertentu, termasuk Jetson Nano. • Greengrass tidak lagi menutup standar komponen di, ini mengembalikan perilaku ke perilaku pra-2.10.0
2.10.0	Fitur baru • Menambahkan `interpolateComponentConfiguration` dukungan untuk ekspresi reguler kosong. Greengrass sekarang diinterpolasi dari objek konfigurasi root. • Menambahkan dukungan untuk MQTT5. • Menambahkan mekanisme untuk memuat komponen plugin dengan cepat tanpa memindai. • Mengaktifkan Greengrass untuk menghemat ruang disk dengan menghapus gambar Docker yang tidak digunakan. Perbaikan bug dan peningkatan • Memperbaiki masalah saat rollback meninggalkan nilai konfigurasi tertentu dari penerapan. • Memperbaiki masalah di mana inti Greengrass memvalidasi urutan domain dalam non-kredensi kustom AWS dan titik akhir data.AWS • Memperbarui resolusi ketergantungan multi-grup untuk menyelesa ikan kembali semua dependensi grup melalui AWS Cloud negosiasi, alih-alih mengunci ke versi aktif. Pembaruan ini juga menghapus kode `INSTALLED_COMPONENT_NOT_FOUND` kesalahan penerapan. • Memperbarui inti Greengrass untuk melewatkan pengunduhan gambar Docker saat sudah ada secara lokal. • Memperbarui inti Greengrass untuk memulai ulang langkah penginstalan komponen sebelum batas waktu berakhir. • Peningkatan dan perbaikan kecil tambahan.

Versi	Perubahan
2.9.6	Perbaikan bug dan peningkatan • Memperbaiki masalah saat penerapan Greengrass gagal dengan kesalahan LAUNCH_DIRECTORY_CORRUPTED dan reboot perangkat berikutnya gagal memulai Greengrass. Kesalahan ini dapat terjadi saat Anda memindahkan perangkat Greengrass di antara beberapa grup hal dengan penerapan yang memerlukan Greengrass untuk memulai ulang.
2.9.5	Fitur baru • Menambahkan dukungan untuk verifikasi tanda tangan perangkat lunak inti Greengrass. Perbaikan bug dan peningkatan • Memperbaiki masalah saat penerapan gagal saat wilayah metadata resep lokal tidak cocok dengan wilayah peluncuran inti Greengrass. Inti Greengrass sekarang bernegosiasi ulang dengan awan ketika ini terjadi. • Memperbaiki masalah saat spooler pesan MQTT terisi dan tidak pernah menghapus pesan. • Peningkatan dan perbaikan kecil tambahan.
2.9.4	Perbaikan bug dan peningkatan • Memeriksa pesan nol sebelum menjatuhkan pesan QOS 0. • Memotong nilai detail status pekerjaan jika melebihi batas karakter 1024. • Memperbarui skrip bootstrap untuk Windows untuk membaca jalur root Greengrass dengan benar jika jalur itu menyertakan spasi. • Pembaruan berlangganan AWS IoT Core sehingga menghapus pesan klien jika respons langganan tidak dikirim. • Memastikan bahwa nukleus memuat konfigurasinya dari file cadangan saat file konfigurasi utama rusak atau hilang.

Versi	Perubahan
2.9.3	Perbaikan bug dan peningkatan • Memastikan ID klien MQTT tidak diduplikasi. • Menambahkan pembacaan dan penulisan file yang lebih kuat untuk menghindari dan memulihkan dari korupsi. • Mencoba ulang gambar docker menarik kesalahan terkait jaringan tertentu. • Menambahkan `noProxyAddresses` opsi untuk koneksi MQTT.
2.9.2	Perbaikan bug dan peningkatan • Memperbaiki masalah saat konfigurasi `interpolateComponentConfiguration` tidak berlaku untuk penerapan yang sedang berlangsung. • Menggunakan OSHI untuk membuat daftar semua proses anak.
2.9.1	Perbaikan bug dan peningkatan • Menambahkan perbaikan di mana Greengrass dimulai ulang jika penerapan menghapus komponen plugin.
2.9.0	Fitur baru • Menambahkan kemampuan untuk membuat subdeployment yang mencoba lagi penerapan dengan subset perangkat yang lebih kecil. Fitur ini menciptakan cara yang lebih efisien untuk menguji dan menyelesaikan penerapan yang gagal. Perbaikan bug dan peningkatan • Meningkatkan dukungan untuk sistem yang tidak memiliki`useradd`,`groupadd`, dan`usermod`. • Peningkatan dan perbaikan kecil tambahan.

Versi	Perubahan
2.8.1	Perbaikan bug dan peningkatan • Memperbaiki masalah saat kode kesalahan penerapan tidak dihasilkan dengan benar dari kesalahan Greengrass API. • Memperbaiki masalah saat pembaruan status armada mengirimkan informasi yang tidak akurat saat komponen mencapai `ERRORED` status selama penerapan. • Memperbaiki masalah di mana penerapan tidak dapat diselesaikan ketika Greengrass memiliki lebih dari 50 langganan yang ada.
2.8.0	Fitur baru • Memperbarui inti Greengrass untuk melaporkan respons status kesehatan penerapan yang mencakup kode kesalahan terperinci ketika ada masalah dalam menerapkan komponen ke perangkat inti. Untuk informasi selengkapnya, lihat Kode kesalahan penyebaran terperinci. • Memperbarui inti Greengrass untuk melaporkan respons status kesehatan komponen yang mencakup kode kesalahan terperinci saat komponen memasuki status atau. `BROKEN` `ERRORED` Untuk informasi selengkapnya, lihat Kode status komponen rinci. • Memperluas bidang pesan status untuk meningkatkan informasi ketersediaan cloud untuk perangkat. • Meningkatkan ketahanan layanan status armada. Perbaikan bug dan peningkatan • Memungkinkan komponen yang rusak untuk menginstal ulang ketika konfigurasinya berubah. • Memperbaiki masalah saat nukleus dimulai ulang selama penerapan bootstrap menyebabkan penerapan gagal. • Memperbaiki masalah di Windows di mana instalasi gagal ketika jalur root berisi spasi. • Memperbaiki masalah saat komponen dimatikan selama penerapan menggunakan skrip shutdown versi baru. • Berbagai peningkatan shutdown. • Peningkatan dan perbaikan kecil tambahan.

Versi	Perubahan
2.7.0	Fitur baru • Memperbarui inti Greengrass untuk mengirim pembaruan status ke cloud saat perangkat AWS IoT Greengrass inti menerapkan penerapan lokal. • Menambahkan dukungan untuk sertifikat klien yang ditandatangani oleh otoritas sertifikat khusus (CA), di mana CA tidak terdaftar AWS IoT. Untuk menggunakan fitur ini, Anda dapat mengatur opsi `greengrasDataPlaneEndpoint` konfigurasi baru `keiotdata`. Untuk informasi selengkapnya, lihat Menggunakan sertifikat perangkat yang ditandatangani oleh CA pribadi. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana inti Greengrass memutar kembali penerapan dalam skenario tertentu saat nukleus dihentikan atau dimulai ulang. Nukleus sekarang melanjutkan penyebaran setelah nukleus dimulai kembali. • Memperbarui penginstal Greengrass untuk menghormati `--start` argumen saat Anda menentukan untuk mengatur perangkat lunak sebagai layanan sistem. • Memperbarui perilaku SubscribeToComponentUpdates untuk menyetel ID penerapan dalam peristiwa di mana nukleus memperbarui komponen. • Peningkatan dan perbaikan kecil tambahan.

Versi	Perubahan
2.6.0	Fitur baru • Menambahkan dukungan untuk wildcard MQTT saat Anda berlangga nan topik penerbitan/berlangganan lokal. Untuk informasi selengkapnya, lihat Pesan lokal publikasi/berlangganan dan SubscribeToTopic. • Menambahkan dukungan untuk variabel resep dalam konfigura si komponen, selain variabel *component_dependen cy_name* `:configuration:` *json_pointer* resep. Anda dapat menggunakan variabel resep ini saat menentukan komponen `DefaultConfiguration` dalam resep atau saat Anda mengonfig urasi komponen dalam penerapan. Untuk mengaktifkan fitur ini, atur opsi ComponentConfiguration konfigurasi interpolasi ke. `true` Untuk informasi selengkapnya, lihat Variabel resep dan Gunakan variabel resep dalam menggabungkan pembaruan. • Menambahkan dukungan penuh untuk kebijakan * otorisasi wildcard in interprocess communication (IPC). Anda sekarang dapat menentuka n * karakter dalam string sumber daya untuk mencocokkan kombinasi karakter apa pun. Untuk informasi selengkapnya, lihat Wildcard dalam kebijakan otorisasi. • Menambahkan dukungan untuk komponen khusus untuk memanggil operasi IPC yang digunakan CLI Greengrass. Anda dapat menggunak an operasi IPC ini untuk mengelola penerapan lokal, melihat detail komponen, dan membuat kata sandi yang dapat Anda gunakan untuk masuk ke konsol debug lokal. Untuk informasi selengkapnya, lihat IPC: Mengelola penerapan dan komponen lokal. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana komponen dependen tidak akan bereaksi ketika dependensi kerasnya memulai ulang atau mengubah status dalam skenario tertentu. • Meningkatkan pesan kesalahan yang dilaporkan perangkat inti ke layanan AWS IoT Greengrass cloud saat penerapan gagal. • Memperbaiki masalah di mana inti Greengrass menerapkan penerapan sesuatu dua kali dalam skenario tertentu saat nukleus dimulai ulang.

Versi	Perubahan
	• Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.
2.5.6	Fitur baru • Menambahkan dukungan untuk modul keamanan perangkat keras yang menggunakan kunci ECC. Anda dapat menggunakan modul keamanan perangkat keras (HSM) untuk menyimpan kunci pribadi dan sertifikat perangkat dengan aman. Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana penerapan tidak pernah selesai saat Anda menerapkan komponen dengan skrip penginstalan yang rusak dalam skenario tertentu. • Meningkatkan kinerja selama startup. • Peningkatan dan perbaikan kecil tambahan.
2.5.5	Fitur baru • Menambahkan variabel `GG_ROOT_CA_PATH` lingkungan untuk komponen, sehingga Anda dapat mengakses sertifikat root certificate authority (CA) di komponen kustom. Perbaikan bug dan peningkatan • Menambahkan dukungan untuk perangkat Windows yang menggunakan bahasa tampilan selain bahasa Inggris. • Memperbarui cara inti Greengrass mem-parsing argumen installer Boolean, sehingga Anda dapat menentukan argumen Boolean tanpa nilai Boolean untuk menentukan nilai. `true` Misalnya, Anda sekarang dapat menentukan `--provision` alih-alih `--provision true` menginstal dengan penyediaan sumber daya otomatis. • Memperbaiki masalah saat perangkat inti tidak melaporkan statusnya ke layanan AWS IoT Greengrass cloud setelah penyediaan dalam skenario tertentu. • Peningkatan dan perbaikan kecil tambahan.

Versi	Perubahan
2.5.4	Perbaikan bug dan peningkatan • Perbaikan bug umum dan perbaikan.
2.5.3	Fitur baru • Menambahkan dukungan untuk integrasi keamanan perangkat keras. Anda dapat menggunakan modul keamanan perangkat keras (HSM) untuk menyimpan kunci pribadi dan sertifikat perangkat dengan aman. Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras. Perbaikan bug dan peningkatan • Memperbaiki masalah dengan pengecualian runtime saat nukleus membuat koneksi MQTT dengan. AWS IoT Core
2.5.2	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana setelah pembaruan inti Greengrass, layanan Windows gagal memulai lagi setelah Anda menghentikannya atau me-reboot perangkat.

Versi	Perubahan
2.5.1	**Warning** Versi ini tidak lagi tersedia. Perbaikan dalam versi ini tersedia di versi yang lebih baru dari komponen ini. Perbaikan bug dan peningkatan • Menambahkan dukungan untuk versi 32-bit dari Java Runtime Environment (JRE) pada Windows. • Mengubah perilaku penghapusan grup untuk perangkat inti yang AWS IoT kebijakannya tidak memberikan `greengrass:ListThingGroupsForCoreDevice` izin. Dengan versi ini, penerapan berlanjut, mencatat peringatan, dan tidak menghapus komponen saat Anda menghapus perangkat inti dari grup sesuatu. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat. • Memperbaiki masalah dengan variabel lingkungan sistem yang disediakan oleh inti Greengrass untuk proses komponen Greengrass. Anda sekarang dapat me-restart komponen untuk menggunakan variabel lingkungan sistem terbaru.

Versi	Perubahan
2.5.0	Fitur baru • Menambahkan dukungan untuk perangkat inti yang menjalankan Windows. • Ubah perilaku penghapusan grup benda. Dengan versi ini, Anda dapat menghapus perangkat inti dari grup benda untuk menghapus komponen grup benda itu di penerapan berikutnya. Sebagai hasil dari perubahan ini, AWS IoT kebijakan perangkat inti harus memiliki `greengrass:ListThingGroupsForCoreDevice` izin. Jika Anda menggunakan penginstal perangkat lunak AWS IoT Greengrass inti untuk menyediakan sumber daya, AWS IoT kebijakan default mengizinkan`greengrass:*` , yang menyertakan izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass. • Menambahkan dukungan untuk konfigurasi proxy HTTPS. Untuk informasi selengkapnya, lihat Hubungkan pada port 443 atau melalui proksi jaringan. • Menambahkan parameter `windowsUser` konfigurasi baru. Anda dapat menggunakan parameter ini untuk menentukan pengguna default yang akan digunakan untuk menjalankan komponen pada perangkat inti Windows. Untuk informasi selengkapnya, lihat Konfigurasikan pengguna yang menjalankan komponen. • Menambahkan opsi `httpClient` konfigurasi baru yang dapat Anda gunakan untuk menyesuaikan batas waktu permintaan HTTP untuk meningkatkan kinerja pada jaringan yang lambat. Untuk informasi selengkapnya, lihat parameter konfigurasi HttpClient. Perbaikan bug dan peningkatan • Memperbaiki opsi siklus hidup bootstrap untuk me-restart perangkat inti dari komponen. • Menambahkan dukungan untuk tanda hubung dalam variabel resep. • Memperbaiki otorisasi IPC untuk komponen fungsi Lambda sesuai permintaan.

Versi	Perubahan
	• Meningkatkan pesan log dan mengubah log non-kritis dari `DEBUG` tingkat `INFO` ke tingkat, sehingga log lebih berguna. • Menghapus `iot:DescribeCertificate` izin dari peran pertukaran token default yang dibuat oleh inti Greengrass saat Anda menginstal AWS IoT Greengrass perangkat lunak Core dengan penyediaan otomatis. Izin ini tidak digunakan oleh inti Greengrass. • Memperbaiki masalah sehingga skrip penyediaan otomatis tidak memerlukan `iam:GetPolicy` izin jika tersedia untuk kebijakan `iam:CreatePolicy` yang sama. • Peningkatan dan perbaikan kecil tambahan.

Versi	Perubahan
2.4.0	Fitur baru • Menambahkan dukungan untuk batas sumber daya sistem. Anda dapat mengonfigurasi jumlah maksimum penggunaan CPU dan RAM yang dapat digunakan oleh setiap proses komponen pada perangkat inti. Untuk informasi selengkapnya, lihat Konfigurasikan batas sumber daya sistem untuk komponen. • Menambahkan operasi IPC untuk menjeda dan melanjutkan komponen. Untuk informasi selengkapnya, lihat PauseComponent dan ResumeComponent. • Menambahkan dukungan untuk penyediaan plugin. Anda dapat menentukan file JAR untuk dijalankan selama instalasi untuk menyediakan AWS sumber daya yang diperlukan untuk perangkat inti Greengrass. Inti Greengrass menyertakan antarmuka yang dapat Anda terapkan untuk mengembangkan plugin penyediaan khusus. Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya khusus. • Menambahkan `thing-name-policy` argumen opsional ke penginstal perangkat lunak AWS IoT Greengrass Core. Anda dapat menggunakan opsi ini untuk menentukan AWS IoT kebijakan yang ada atau kustom saat Anda menginstal perangkat lunak AWS IoT Greengrass Inti dengan penyediaan sumber daya otomatis. Perbaikan bug dan peningkatan • Memperbarui konfigurasi logging saat startup. Ini memperbaiki masalah di mana konfigurasi logging tidak diterapkan saat startup. • Memperbarui symlink pemuat nukleus untuk menunjuk ke penyimpanan komponen di folder root Greengrass selama penginstalan. Pembaruan ini memungkinkan Anda untuk menghapus file JAR dan artefak inti lainnya yang Anda unduh saat Anda menginstal perangkat lunak AWS IoT Greengrass Core. • Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.

Versi	Perubahan
2.3.0	Fitur baru • Menambahkan dukungan untuk dokumen konfigurasi deployment hingga 10 MB, naik dari 7 KB (untuk deployment yang menargetkan objek) atau 31 KB (untuk deployment yang menargetkan objek grup). Untuk menggunakan fitur ini, AWS IoT kebijakan perangkat inti harus mengizinkan `greengrass:GetDeploymentConfiguration` izin. Jika Anda menggunakan penginstal perangkat lunak AWS IoT Greengrass inti untuk menyediakan sumber daya, AWS IoT kebijakan perangkat inti Anda mengizinkan`greengrass:*` , yang mencakup izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass. • Menambahkan variabel resep `iot:thingName` . Anda dapat menggunakan variabel resep ini untuk mendapatkan nama perangkat AWS IoT inti dalam resep. Untuk informasi selengkapnya, lihat Variabel resep. Perbaikan bug dan peningkatan • Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.
2.2.0	Fitur baru • Menambahkan operasi IPC untuk manajemen bayangan lokal. Perbaikan bug dan peningkatan • Mengurangi ukuran file JAR. • Mengurangi penggunaan memori. • Memperbaiki masalah di mana konfigurasi log tidak diperbarui dalam kasus tertentu. • Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.

Versi	Perubahan

2.1.0	Fitur baru
	• Mendukung pengunduhan gambar Docker dari repositori privat di Amazon ECR.
	• Menambahkan parameter berikut untuk menyesuaikan konfigurasi MQTT pada perangkat inti:
	  • `maxInFlightPublishes` - Jumlah maksimum pesan MQTT QoS 1 yang tidak diakui yang dapat terbang pada waktu yang sama.
	  • `maxPublishRetry` - Jumlah waktu maksimum untuk mencoba kembali pesan yang gagal untuk dipublikasikan.
	• Menambahkan parameter konfigurasi `fleetstatusservice` untuk mengonfigurasi interval di mana perangkat inti menerbitkan status perangkat untuk AWS Cloud.
	• Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.
	Perbaikan bug dan peningkatan
	• Memperbaiki masalah yang menyebabkan deployment bayangan diduplikasi saat nukleus dimulai ulang.
	• Memperbaiki masalah yang menyebabkan nukleus lumpuh saat mengalami pengecualian beban layanan.
	• Meningkatkan resolusi dependensi komponen untuk menggagalkan deployment yang mencakup dependensi melingkar.
	• Memperbaiki masalah yang mencegah komponen plugin di-deploy ulang jika komponen tersebut sebelumnya telah dihapus dari perangkat inti.
	• Memperbaiki masalah yang menyebabkan variabel lingkungan `HOME` yang akan ditetapkan ke direktori */greengrass/v2* `/work` untuk komponen Lambda atau untuk komponen yang berjalan sebagai root. Variabel `HOME` sekarang diatur dengan tepat ke direktori home untuk pengguna yang menjalankan komponen.
	• Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.

Versi	Perubahan
2.0.5	Perbaikan bug dan peningkatan • Rutekan lalu lintas dengan benar melalui proxy jaringan yang dikonfigu rasi saat mengunduh komponen AWS yang disediakan. • Menggunakan titik akhir bidang data Greengrass yang benar di Wilayah Cina AWS .
2.0.4	Fitur baru • Memungkinkan lalu lintas HTTPS melalui port 443. Anda dapat menggunakan parameter konfigurasi `greengrassDataPlanePort` untuk versi 2.0.4 komponen inti untuk mengonfigurasi komunikasi HTTPS untuk berjalan melalui port 443 dan bukan port default 8443. Untuk informasi selengkapnya, lihat Konfigurasikan HTTPS melalui port 443. • Menambahkan variabel resep jalur kerja. Anda dapat menggunakan variabel resep ini untuk mendapatkan jalur ke folder kerja komponen, yang dapat Anda gunakan untuk berbagi file antara komponen dan dependensinya. Untuk informasi lebih lanjut, lihat variabel resep jalur kerja. Perbaikan bug dan peningkatan • Mencegah pembuatan kebijakan peran pertukaran token AWS Identity and Access Management (IAM) jika kebijakan peran sudah ada. Akibat dari perubahan ini, penginstal sekarang memerlukan `iam:GetPolicy` dan `sts:GetCallerIdentity` ketika dijalankan dengan `--provision true` . Untuk informasi selengkapnya, lihat Kebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya. • Secara tepat menangani pembatalan deployment yang belum berhasil didaftarkan. • Meng-update konfigurasi untuk menghapus entri lama dengan cap waktu yang lebih baru ketika memutar kembali deployment. • Peningkatan dan perbaikan kecil tambahan. Untuk informasi lebih lanjut, lihat rilis di GitHub.

Versi	Perubahan
2.0.3	Versi awal.

Auth perangkat klien

Komponen auth perangkat klien (`aws.greengrass.clientdevices.Auth`) mengautentikasi perangkat klien dan mengotorisasi tindakan perangkat klien.

> **Note**
>
> Perangkat klien adalah perangkat IoT lokal yang terhubung ke perangkat inti Greengrass untuk mengirim pesan MQTT dan data yang akan diproses. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Topik

Versi

> **Note**
>
> Autentikasi perangkat klien versi 2.3.0 telah dihentikan. Kami sangat menyarankan Anda meningkatkan ke autentikasi perangkat klien versi 2.3.1 atau yang lebih baru.

Komponen ini memiliki versi berikut:

- 2.4.x
- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Jenis

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Peran layanan Greengrass harus dikaitkan dengan Anda dan mengizinkan Akun AWS izin. `iot:DescribeCertificate`
- AWS IoT Kebijakan perangkat inti harus mengizinkan izin berikut:
  - `greengrass:GetConnectivityInfo`, di mana sumber daya termasuk ARN perangkat inti yang menjalankan komponen ini
  - `greengrass:VerifyClientDeviceIoTCertificateAssociation`, di mana sumber daya menyertakan Nama Sumber Daya Amazon (ARN) dari setiap perangkat klien yang terhubung ke perangkat inti

- `greengrass:VerifyClientDeviceIdentity`
- `greengrass:PutCertificateAuthorities`
- `iot:Publish`, di mana sumber daya termasuk ARN dari topik MQTT berikut:
  - `$aws/things/coreDeviceThingName*-gci/shadow/get`
- `iot:Subscribe`, di mana sumber daya mencakup ARN dari filter topik MQTT berikut:
  - `$aws/things/coreDeviceThingName*-gci/shadow/update/delta`
  - `$aws/things/coreDeviceThingName*-gci/shadow/get/accepted`
- `iot:Receive`, di mana sumber daya mencakup ARN dari topik MQTT berikut:
  - `$aws/things/coreDeviceThingName*-gci/shadow/update/delta`
  - `$aws/things/coreDeviceThingName*-gci/shadow/get/accepted`

  Untuk informasi selengkapnya, lihat Kebijakan AWS IoT untuk operasi bidang data dan Kebijakan AWS IoT minimal untuk mendukung perangkat klien.
- (Opsional) Untuk menggunakan otentikasi offline, peran AWS Identity and Access Management (IAM) yang digunakan oleh AWS IoT Greengrass layanan harus berisi izin berikut:
  - `greengrass:ListClientDevicesAssociatedWithCoreDevice`untuk mengaktifkan perangkat inti untuk daftar klien untuk otentikasi offline.
- Komponen autentikasi perangkat klien didukung untuk berjalan di VPC. Untuk menerapkan komponen ini di VPC, berikut ini diperlukan.
  - Komponen autentikasi perangkat klien harus memiliki konektivitas ke AWS IoT data, AWS IoT Kredensyal, dan Amazon S3.

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Wajib	Deskripsi
`iot.region.amazonaws.com`	443	Ya	Digunakan untuk mendapatkan

Titik Akhir	Port	Wajib	Deskripsi
			informasi tentang sertifikat AWS IoT barang.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.4.4

Tabel berikut mencantumkan dependensi untuk versi 2.4.4 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.6.0 <2.13.0	Lunak

2.4.3

Tabel berikut mencantumkan dependensi untuk versi 2.4.3 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.6.0 <2.12.0	Lunak

2.4.1 and 2.4.2

Tabel berikut mencantumkan dependensi untuk versi 2.4.1 dan 2.4.2 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.6.0 <2.11.0	Lunak

2.3.0 – 2.4.0

Tabel berikut mencantumkan dependensi untuk versi 2.3.0 hingga 2.4.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.6.0 <2.10.0	Lunak

2.3.0

Tabel berikut mencantumkan dependensi untuk versi 2.3.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.6.0 <2.10.0	Lunak

2.2.3

Tabel berikut mencantumkan dependensi untuk versi 2.2.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.6.0 <=2.9.0	Lunak

2.2.2

Tabel berikut mencantumkan dependensi untuk versi 2.2.2 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.6.0 <=2.8.0	Lunak

2.2.1

Tabel berikut mencantumkan dependensi untuk versi 2.2.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.6.0 <2.8.0	Lunak

2.2.0

Tabel berikut mencantumkan dependensi untuk versi 2.2.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.6.0 <2.7.0	Lunak

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.7.0	Lunak

2.0.4

Tabel berikut mencantumkan dependensi untuk versi 2.0.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.6.0	Lunak

2.0.2 and 2.0.3

Tabel berikut mencantumkan dependensi untuk versi 2.0.2 dan 2.0.3 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.5.0	Lunak

2.0.1

Tabel berikut mencantumkan dependensi untuk versi 2.0.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.4.0	Lunak

2.0.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.3.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

> **Note**
>
> Izin berlangganan dievaluasi selama permintaan berlangganan klien ke broker MQTT lokal. Jika izin berlangganan klien yang ada dicabut, klien tidak akan lagi dapat berlangganan topik. Namun, itu akan terus menerima pesan dari topik berlangganan sebelumnya. Untuk mencegah perilaku ini, broker MQTT lokal harus dimulai kembali setelah mencabut izin berlangganan untuk memaksa otorisasi ulang klien.
> Untuk komponen broker MQTT 5 (EMQX), perbarui `restartIdentifier` konfigurasi untuk memulai kembali broker MQTT 5. Untuk informasi lebih lanjut, lihat konfigurasi komponen broker MQTT 5.

Untuk komponen broker MQTT 3.1.1 (Moquette), komponen ini dimulai ulang setiap minggu secara default ketika sertifikat server berubah memaksa klien untuk mengotorisasi ulang. Anda dapat memaksa restart baik dengan mengubah informasi konektivitas (alamat IP) dari perangkat inti atau dengan membuat penyebaran untuk menghapus komponen broker dan kemudian menerapkannya lagi nanti.

v2.5.0

`deviceGroups`

Grup perangkat adalah grup perangkat klien yang memiliki izin untuk menyambung dan berkomunikasi dengan perangkat inti. Gunakan aturan pemilihan untuk mengidentifikasi grup perangkat klien, dan menentukan kebijakan otorisasi perangkat klien yang menentukan izin untuk setiap grup perangkat.

Objek ini berisi informasi berikut:

`formatVersion`

Versi format untuk objek konfigurasi ini.

Pilih dari salah satu pilihan berikut:

- `2021-03-05`

`definitions`

grup perangkat untuk perangkat inti ini. Setiap definisi menentukan aturan pemilihan untuk mengevaluasi apakah perangkat klien adalah anggota grup. Setiap definisi juga menentukan kebijakan izin yang akan diterapkan ke perangkat klien yang cocok dengan aturan pemilihan. Jika perangkat klien adalah anggota dari beberapa grup perangkat, izin perangkat terdiri dari kebijakan izin masing-masing grup.

Objek ini berisi informasi berikut:

`groupNameKey`

Nama grup perangkat ini. Ganti *`groupNameKey`*dengan nama yang membantu Anda mengidentifikasi grup perangkat ini.

Objek ini berisi informasi berikut:

`selectionRule`

Kueri yang menentukan perangkat klien mana yang menjadi anggota grup perangkat ini. Saat perangkat klien terhubung, perangkat inti mengevaluasi aturan pemilihan ini untuk menentukan apakah perangkat klien adalah anggota grup perangkat ini. Jika perangkat klien adalah anggota, perangkat inti akan menggunakan kebijakan grup perangkat ini untuk mengotorisasi tindakan perangkat klien.

Setiap aturan pemilihan terdiri dari setidaknya satu klausa aturan pemilihan, yang merupakan kueri ekspresi tunggal yang dapat mencocokkan perangkat klien. Aturan pemilihan menggunakan sintaks kueri yang sama dengan pengindeksan AWS IoT armada. Untuk informasi selengkapnya tentang sintaks aturan pemilihan, lihat sintaks kueri pengindeksan AWS IoT armada di Panduan Pengembang.AWS IoT Core

Gunakan wildcard * untuk mencocokkan beberapa perangkat klien dengan satu pilihan klausul aturan. Anda dapat menggunakan wildcard ini di awal dan akhir nama benda untuk mencocokkan perangkat klien yang namanya dimulai atau diakhiri dengan string yang Anda tentukan. Anda juga dapat menggunakan wildcard ini untuk mencocokkan semua perangkat klien.

> **Note**
>
> Untuk memilih nilai yang berisi karakter titik dua (`:`), lepaskan titik dua dengan karakter garis miring terbalik (\). Dalam format seperti JSON, Anda harus melepaskan karakter ris miring terbalik, sehingga Anda memasukkan dua karakter ris miring terbalik sebelum karakter titik dua. Sebagai contoh, tentukan `thingName: MyTeam\\:ClientDevice1` untuk memilih objek yang namanya `MyTeam:ClientDevice1`.

Anda dapat menentukan sebagai berikut:

- `thingName`— Nama AWS IoT benda perangkat klien.

Example Contoh aturan pemilihan

Aturan seleksi berikut cocok dengan perangkat klien yang namanya `MyClientDevice1` atau `MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Contoh aturan seleksi (gunakan wildcard)

Aturan seleksi berikut cocok dengan perangkat klien yang namanya dimulai dengan `MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Contoh aturan seleksi (gunakan wildcard)

Aturan pemilihan berikut cocok dengan perangkat klien yang namanya diakhiri dengan`MyClientDevice`.

```
thingName: *MyClientDevice
```

Example Contoh aturan pemilihan (cocok dengan semua perangkat)

Aturan seleksi berikut cocok dengan semua perangkat klien.

```
thingName: *
```

`policyName`

Kebijakan izin yang berlaku untuk perangkat klien dalam grup perangkat ini. Tentukan nama kebijakan yang Anda tetapkan di objek `policies`.

`policies`

Kebijakan otorisasi perangkat klien untuk perangkat klien yang terhubung ke perangkat inti. Setiap kebijakan otorisasi menentukan serangkaian tindakan dan sumber daya tempat perangkat klien dapat melakukan tindakan tersebut.

Objek ini berisi informasi berikut:

`policyNameKey`

Nama kebijakan otorisasi ini. Ganti *`policyNameKey`*dengan nama yang membantu Anda mengidentifikasi kebijakan otorisasi ini. Anda menggunakan nama kebijakan ini untuk menentukan kebijakan yang berlaku untuk grup perangkat.

Objek ini berisi informasi berikut:

statementNameKey

Nama pernyataan kebijakan ini. Ganti *statementNameKey*dengan nama yang membantu Anda mengidentifikasi pernyataan kebijakan ini.

Objek ini berisi informasi berikut:

operations

Daftar operasi untuk mengizinkan sumber daya dalam kebijakan ini.

Anda dapat menyertakan salah satu dari operasi berikut:

- `mqtt:connect` — Memberikan izin untuk terhubung ke perangkat inti. Perangkat klien harus memiliki izin ini untuk menyambung ke perangkat inti.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:clientId:`*deviceClientId* — Batasi akses berdasarkan ID klien yang digunakan perangkat klien untuk terhubung ke broker MQTT perangkat inti. Ganti *deviceClientId*dengan ID klien yang akan digunakan.
- `mqtt:publish` — Memberikan izin untuk mempublikasikan pesan MQTT ke topik.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topic:`*mqttTopic* — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *MqttTopic* dengan topik yang akan digunakan.

    Sumber daya ini tidak mendukung wildcard topik MQTT.
- `mqtt:subscribe` — Memberikan izin untuk berlangganan filter topik MQTT untuk menerima pesan.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topicfilter:`*mqttTopicFilter* — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *mqttTopicFilter*dengan filter topik yang akan digunakan.

    Sumber daya ini mendukung wildcard topik MQTT + dan #. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

Perangkat klien dapat berlangganan filter topik yang tepat yang Anda izinkan. Misalnya, jika Anda mengizinkan perangkat klien untuk berlangganan sumber daya `mqtt:topicfilter:client/+/status`, perangkat klien dapat berlangganan `client/+/status` tetapi bukan `client/client1/status`.

Anda dapat menentukan wildcard `*` untuk mengizinkan akses ke semua tindakan.

`resources`

Daftar operasi yang akan mengizinkan sumber daya dalam kebijakan ini. Tentukan sumber daya yang sesuai dengan operasi dalam kebijakan ini. Misalnya, Anda dapat menentukan daftar sumber daya topik MQTT (`mqtt:topic:`*`mqttTopic`*) dalam kebijakan yang menentukan operasi `mqtt:publish`.

Anda dapat menentukan `*` wildcard di mana saja dalam variabel sumber daya untuk memungkinkan akses ke semua sumber daya. Misalnya, Anda dapat menentukan **`mqtt:topic:my*`** untuk mengizinkan akses ke sumber daya yang cocok dengan input tersebut.

Variabel sumber daya berikut didukung:

- `mqtt:topic:${iot:Connection.Thing.ThingName}`

  Ini menyelesaikan nama benda dalam AWS IoT Core registri tempat kebijakan sedang dievaluasi. AWS IoT Core menggunakan sertifikat yang disajikan perangkat saat mengautentikasi untuk menentukan hal mana yang akan digunakan untuk memverifikasi koneksi. Variabel kebijakan ini hanya tersedia jika perangkat terhubung melalui MQTT atau MQTT melalui protokol. WebSocket

`statementDescription`

(Opsional) Deskripsi untuk pernyataan kebijakan ini.

`certificates`

(Opsional) Opsi konfigurasi sertifikat untuk perangkat inti ini. Objek ini berisi informasi berikut:

`serverCertificateValiditySeconds`

(Opsional) Jumlah waktu (dalam detik) setelah sertifikat server MQTT lokal kedaluwarsa. Anda dapat mengonfigurasi opsi ini untuk menyesuaikan seberapa sering perangkat klien memutuskan sambungan dan menyambung kembali ke perangkat inti.

Komponen ini memutar sertifikat server MQTT lokal 24 jam sebelum kedaluwarsa. Broker MQTT, seperti komponen broker Moquette MQTT, menghasilkan sertifikat baru dan memulai ulang. Ketika ini terjadi, semua perangkat klien yang terhubung ke perangkat inti ini terputus. Perangkat klien dapat terhubung kembali ke perangkat inti setelah periode waktu yang singkat.

Default: `604800` (7 hari)

Nilai minimum: `172800` (2 hari)

Nilai maksimum: `864000` (10 hari)

`performance`

(Opsional) Opsi konfigurasi kinerja untuk perangkat inti ini. Objek ini berisi informasi berikut:

`maxActiveAuthTokens`

(Opsional) Jumlah maksimum token otorisasi perangkat klien aktif. Anda dapat meningkatkan jumlah ini untuk mengaktifkan lebih banyak perangkat klien untuk terhubung ke perangkat inti tunggal, tanpa mengautentikasi ulang mereka.

Default: `2500`

`cloudRequestQueueSize`

(Opsional) Jumlah maksimum AWS Cloud permintaan untuk mengantri sebelum komponen ini menolak permintaan.

Default: `100`

`maxConcurrentCloudRequests`

(Opsional) Jumlah maksimum permintaan bersamaan untuk dikirim ke. AWS Cloud Anda dapat meningkatkan angka ini untuk meningkatkan kinerja otentikasi pada perangkat inti tempat Anda menghubungkan sejumlah besar perangkat klien.

Default: 1

`certificateAuthority`

(Opsional) Opsi konfigurasi otoritas sertifikat untuk mengganti otoritas perantara perangkat inti dengan otoritas sertifikat perantara Anda sendiri.

> **Note**
>
> Jika Anda mengonfigurasi perangkat inti Greengrass Anda dengan otoritas sertifikat khusus (CA) dan menggunakan CA yang sama untuk menerbitkan sertifikat perangkat klien, Greengrass melewati pemeriksaan kebijakan otorisasi untuk operasi MQTT perangkat klien. Komponen autentikasi perangkat klien sepenuhnya mempercayai klien menggunakan sertifikat yang ditandatangani oleh CA yang dikonfigurasi untuk digunakan.
> Untuk membatasi perilaku ini saat menggunakan CA kustom, buat dan tandatangani perangkat klien menggunakan CA atau CA perantara yang berbeda, lalu sesuaikan `certificateChainUri` bidang `certificateUri` dan untuk menunjuk ke CA perantara yang benar.

Objek ini berisi informasi berikut.

CertificateURI

Lokasi sertifikat. Ini bisa berupa URI sistem file atau URI yang menunjuk ke sertifikat yang disimpan dalam modul keamanan perangkat keras.

`certificateChainUri`

Lokasi rantai sertifikat untuk perangkat inti CA. Ini harus menjadi rantai sertifikat lengkap kembali ke CA root Anda. Ini bisa berupa URI sistem file atau URI yang menunjuk ke rantai sertifikat yang disimpan dalam modul keamanan perangkat keras.

`privateKeyUri`

Lokasi kunci pribadi perangkat inti. Ini bisa berupa URI sistem file atau URI yang menunjuk ke kunci pribadi sertifikat yang disimpan dalam modul keamanan perangkat keras.

`security`

(Opsional) Opsi konfigurasi keamanan untuk perangkat inti ini. Objek ini berisi informasi berikut.

`clientDeviceTrustDurationMinutes`

Durasi dalam hitungan menit bahwa informasi otentikasi perangkat klien dapat dipercaya sebelum diperlukan untuk mengautentikasi ulang dengan perangkat inti. Nilai default adalah 1.

`metrics`

(Opsional) Opsi metrik untuk perangkat inti ini. Metrik kesalahan hanya akan ditampilkan jika ada kesalahan dengan autentikasi perangkat klien. Objek ini berisi informasi berikut:

`disableMetrics`

Jika `disableMetrics` bidang ditetapkan sebagai`true`, autentikasi perangkat klien tidak akan mengumpulkan metrik.

Default: `false`

`aggregatePeriodSeconds`

Periode agregasi dalam hitungan detik yang menentukan seberapa sering autentikasi perangkat klien mengumpulkan metrik dan mengirimkannya ke agen telemetri. Ini tidak mengubah seberapa sering metrik diterbitkan karena agen telemetri masih menerbitkannya sekali sehari.

Default: `3600`

startupTimeoutSeconds

(Opsional) Maksimum waktu dalam hitungan detik untuk memulai komponen. Status komponen berubah menjadi `BROKEN` jika melebihi batas waktu ini.

Default: `120`

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan yang ketat)

Contoh konfigurasi berikut menentukan untuk memungkinkan perangkat klien yang namanya dimulai dengan `MyClientDevice` untuk menyambung dan mempublikasikan/berlangganan semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
```

```
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
          ]
        }
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan permisif)

Contoh konfigurasi berikut menentukan untuk memungkinkan semua perangkat klien untuk terhubung dan mempublikasikan/berlangganan pada semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan nama benda)

Contoh konfigurasi berikut memungkinkan perangkat klien untuk mempublikasikan topik yang dimulai dengan nama benda perangkat klien dan diakhiri dengan string`topic`.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "myThing": {
        "selectionRule": "thingName: *",
        "policyName": "MyThingNamePolicy"
      }
```

```
    },
    "policies": {
      "MyThingNamePolicy": {
        "policyStatement": {
          "statementDescription": "mqtt publish",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:${iot:Connection.Thing.ThingName}/*/topic"
          ]
        }
      }
    }
  }
}
```

v2.4.5

`deviceGroups`

Grup perangkat adalah grup perangkat klien yang memiliki izin untuk menyambung dan berkomunikasi dengan perangkat inti. Gunakan aturan pemilihan untuk mengidentifikasi grup perangkat klien, dan menentukan kebijakan otorisasi perangkat klien yang menentukan izin untuk setiap grup perangkat.

Objek ini berisi informasi berikut:

`formatVersion`

Versi format untuk objek konfigurasi ini.

Pilih dari salah satu pilihan berikut:

- `2021-03-05`

`definitions`

grup perangkat untuk perangkat inti ini. Setiap definisi menentukan aturan pemilihan untuk mengevaluasi apakah perangkat klien adalah anggota grup. Setiap definisi juga menentukan kebijakan izin yang akan diterapkan ke perangkat klien yang cocok dengan aturan pemilihan. Jika perangkat klien adalah anggota dari beberapa grup perangkat, izin perangkat terdiri dari kebijakan izin masing-masing grup.

Objek ini berisi informasi berikut:

groupNameKey

Nama grup perangkat ini. Ganti *groupNameKey*dengan nama yang membantu Anda mengidentifikasi grup perangkat ini.

Objek ini berisi informasi berikut:

`selectionRule`

Kueri yang menentukan perangkat klien mana yang menjadi anggota grup perangkat ini. Saat perangkat klien terhubung, perangkat inti mengevaluasi aturan pemilihan ini untuk menentukan apakah perangkat klien adalah anggota grup perangkat ini. Jika perangkat klien adalah anggota, perangkat inti akan menggunakan kebijakan grup perangkat ini untuk mengotorisasi tindakan perangkat klien.

Setiap aturan pemilihan terdiri dari setidaknya satu klausa aturan pemilihan, yang merupakan kueri ekspresi tunggal yang dapat mencocokkan perangkat klien. Aturan pemilihan menggunakan sintaks kueri yang sama dengan pengindeksan AWS IoT armada. Untuk informasi selengkapnya tentang sintaks aturan pemilihan, lihat sintaks kueri pengindeksan AWS IoT armada di Panduan Pengembang.AWS IoT Core

Gunakan wildcard `*` untuk mencocokkan beberapa perangkat klien dengan satu pilihan klausul aturan. Anda dapat menggunakan wildcard ini di awal dan akhir nama benda untuk mencocokkan perangkat klien yang namanya dimulai atau diakhiri dengan string yang Anda tentukan. Anda juga dapat menggunakan wildcard ini untuk mencocokkan semua perangkat klien.

> **Note**
>
> Untuk memilih nilai yang berisi karakter titik dua (`:`), lepaskan titik dua dengan karakter garis miring terbalik (\). Dalam format seperti JSON, Anda harus melepaskan karakter ris miring terbalik, sehingga Anda memasukkan dua karakter ris miring terbalik sebelum karakter titik dua. Sebagai contoh, tentukan `thingName: MyTeam\\:ClientDevice1` untuk memilih objek yang namanya `MyTeam:ClientDevice1`.

Anda dapat menentukan sebagai berikut:

- `thingName`— Nama AWS IoT benda perangkat klien.

Example Contoh aturan pemilihan

Aturan seleksi berikut cocok dengan perangkat klien yang namanya `MyClientDevice1` atau `MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Contoh aturan seleksi (gunakan wildcard)

Aturan seleksi berikut cocok dengan perangkat klien yang namanya dimulai dengan `MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Contoh aturan seleksi (gunakan wildcard)

Aturan pemilihan berikut cocok dengan perangkat klien yang namanya diakhiri dengan`MyClientDevice`.

```
thingName: *MyClientDevice
```

Example Contoh aturan pemilihan (cocok dengan semua perangkat)

Aturan seleksi berikut cocok dengan semua perangkat klien.

```
thingName: *
```

`policyName`

Kebijakan izin yang berlaku untuk perangkat klien dalam grup perangkat ini. Tentukan nama kebijakan yang Anda tetapkan di objek `policies`.

`policies`

Kebijakan otorisasi perangkat klien untuk perangkat klien yang terhubung ke perangkat inti. Setiap kebijakan otorisasi menentukan serangkaian tindakan dan sumber daya tempat perangkat klien dapat melakukan tindakan tersebut.

Objek ini berisi informasi berikut:

policyNameKey

Nama kebijakan otorisasi ini. Ganti *policyNameKey*dengan nama yang membantu Anda mengidentifikasi kebijakan otorisasi ini. Anda menggunakan nama kebijakan ini untuk menentukan kebijakan yang berlaku untuk grup perangkat.

Objek ini berisi informasi berikut:

statementNameKey

Nama pernyataan kebijakan ini. Ganti *statementNameKey*dengan nama yang membantu Anda mengidentifikasi pernyataan kebijakan ini.

Objek ini berisi informasi berikut:

`operations`

Daftar operasi untuk mengizinkan sumber daya dalam kebijakan ini.

Anda dapat menyertakan salah satu dari operasi berikut:

- `mqtt:connect` — Memberikan izin untuk terhubung ke perangkat inti. Perangkat klien harus memiliki izin ini untuk menyambung ke perangkat inti.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:clientId:`*`deviceClientId`* — Batasi akses berdasarkan ID klien yang digunakan perangkat klien untuk terhubung ke broker MQTT perangkat inti. Ganti *deviceClientId*dengan ID klien yang akan digunakan.
- `mqtt:publish` — Memberikan izin untuk mempublikasikan pesan MQTT ke topik.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topic:`*`mqttTopic`* — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *MqttTopic* dengan topik yang akan digunakan.

    Sumber daya ini tidak mendukung wildcard topik MQTT.
- `mqtt:subscribe` — Memberikan izin untuk berlangganan filter topik MQTT untuk menerima pesan.

  Operasi ini mendukung sumber daya berikut:

- `mqtt:topicfilter:`*`mqttTopicFilter`* — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`mqttTopicFilter`*dengan filter topik yang akan digunakan.

  Sumber daya ini mendukung wildcard topik MQTT + dan #. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

  Perangkat klien dapat berlangganan filter topik yang tepat yang Anda izinkan. Misalnya, jika Anda mengizinkan perangkat klien untuk berlangganan sumber daya `mqtt:topicfilter:client/+/status`, perangkat klien dapat berlangganan `client/+/status` tetapi bukan `client/client1/status`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua tindakan.

`resources`

Daftar operasi yang akan mengizinkan sumber daya dalam kebijakan ini. Tentukan sumber daya yang sesuai dengan operasi dalam kebijakan ini. Misalnya, Anda dapat menentukan daftar sumber daya topik MQTT (`mqtt:topic:`*`mqttTopic`*) dalam kebijakan yang menentukan operasi `mqtt:publish`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua sumber daya. Anda tidak dapat menggunakan wildcard * untuk mencocokkan pengidentifikasi sumber daya parsial. Misalnya, Anda dapat menentukan **`"resources": "*"`**, tetapi Anda tidak dapat menentukan **`"resources": "mqtt:clientId:*"`**.

`statementDescription`

(Opsional) Deskripsi untuk pernyataan kebijakan ini.

`certificates`

(Opsional) Opsi konfigurasi sertifikat untuk perangkat inti ini. Objek ini berisi informasi berikut:

`serverCertificateValiditySeconds`

(Opsional) Jumlah waktu (dalam detik) setelah sertifikat server MQTT lokal kedaluwarsa. Anda dapat mengonfigurasi opsi ini untuk menyesuaikan seberapa sering perangkat klien memutuskan sambungan dan menyambung kembali ke perangkat inti.

Komponen ini memutar sertifikat server MQTT lokal 24 jam sebelum kedaluwarsa. Broker MQTT, seperti komponen broker Moquette MQTT, menghasilkan sertifikat baru dan memulai ulang. Ketika ini terjadi, semua perangkat klien yang terhubung ke perangkat inti ini terputus. Perangkat klien dapat terhubung kembali ke perangkat inti setelah periode waktu yang singkat.

Default: `604800` (7 hari)

Nilai minimum: `172800` (2 hari)

Nilai maksimum: `864000` (10 hari)

`performance`

(Opsional) Opsi konfigurasi kinerja untuk perangkat inti ini. Objek ini berisi informasi berikut:

`maxActiveAuthTokens`

(Opsional) Jumlah maksimum token otorisasi perangkat klien aktif. Anda dapat meningkatkan jumlah ini untuk mengaktifkan lebih banyak perangkat klien untuk terhubung ke perangkat inti tunggal, tanpa mengautentikasi ulang mereka.

Default: `2500`

`cloudRequestQueueSize`

(Opsional) Jumlah maksimum AWS Cloud permintaan untuk mengantri sebelum komponen ini menolak permintaan.

Default: `100`

`maxConcurrentCloudRequests`

(Opsional) Jumlah maksimum permintaan bersamaan untuk dikirim ke. AWS Cloud Anda dapat meningkatkan angka ini untuk meningkatkan kinerja otentikasi pada perangkat inti tempat Anda menghubungkan sejumlah besar perangkat klien.

Default: 1

`certificateAuthority`

(Opsional) Opsi konfigurasi otoritas sertifikat untuk mengganti otoritas perantara perangkat inti dengan otoritas sertifikat perantara Anda sendiri.

> **Note**
>
> Jika Anda mengonfigurasi perangkat inti Greengrass Anda dengan otoritas sertifikat khusus (CA) dan menggunakan CA yang sama untuk menerbitkan sertifikat perangkat klien, Greengrass melewati pemeriksaan kebijakan otorisasi untuk operasi MQTT perangkat klien. Komponen autentikasi perangkat klien sepenuhnya mempercayai klien menggunakan sertifikat yang ditandatangani oleh CA yang dikonfigurasi untuk digunakan.
> Untuk membatasi perilaku ini saat menggunakan CA kustom, buat dan tandatangani perangkat klien menggunakan CA atau CA perantara yang berbeda, lalu sesuaikan `certificateChainUri` bidang `certificateUri` dan untuk menunjuk ke CA perantara yang benar.

Objek ini berisi informasi berikut.

CertificateURI

Lokasi sertifikat. Ini bisa berupa URI sistem file atau URI yang menunjuk ke sertifikat yang disimpan dalam modul keamanan perangkat keras.

`certificateChainUri`

Lokasi rantai sertifikat untuk perangkat inti CA. Ini harus menjadi rantai sertifikat lengkap kembali ke CA root Anda. Ini bisa berupa URI sistem file atau URI yang menunjuk ke rantai sertifikat yang disimpan dalam modul keamanan perangkat keras.

`privateKeyUri`

Lokasi kunci pribadi perangkat inti. Ini bisa berupa URI sistem file atau URI yang menunjuk ke kunci pribadi sertifikat yang disimpan dalam modul keamanan perangkat keras.

`security`

(Opsional) Opsi konfigurasi keamanan untuk perangkat inti ini. Objek ini berisi informasi berikut.

`clientDeviceTrustDurationMinutes`

Durasi dalam hitungan menit bahwa informasi otentikasi perangkat klien dapat dipercaya sebelum diperlukan untuk mengautentikasi ulang dengan perangkat inti. Nilai default adalah 1.

`metrics`

(Opsional) Opsi metrik untuk perangkat inti ini. Metrik kesalahan hanya akan ditampilkan jika ada kesalahan dengan autentikasi perangkat klien. Objek ini berisi informasi berikut:

`disableMetrics`

Jika `disableMetrics` bidang ditetapkan sebagai`true`, autentikasi perangkat klien tidak akan mengumpulkan metrik.

Default: `false`

`aggregatePeriodSeconds`

Periode agregasi dalam hitungan detik yang menentukan seberapa sering autentikasi perangkat klien mengumpulkan metrik dan mengirimkannya ke agen telemetri. Ini tidak mengubah seberapa sering metrik diterbitkan karena agen telemetri masih menerbitkannya sekali sehari.

Default: `3600`

startupTimeoutSeconds

(Opsional) Maksimum waktu dalam hitungan detik untuk memulai komponen. Status komponen berubah menjadi `BROKEN` jika melebihi batas waktu ini.

Default: `120`

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan yang ketat)

Contoh konfigurasi berikut menentukan untuk memungkinkan perangkat klien yang namanya dimulai dengan `MyClientDevice` untuk menyambung dan mempublikasikan/berlangganan semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
```

```
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
          ]
        }
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan permisif)

Contoh konfigurasi berikut menentukan untuk memungkinkan semua perangkat klien untuk terhubung dan mempublikasikan/berlangganan pada semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
```

```
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

v2.4.2 - v2.4.4

`deviceGroups`

Grup perangkat adalah grup perangkat klien yang memiliki izin untuk menyambung dan berkomunikasi dengan perangkat inti. Gunakan aturan pemilihan untuk mengidentifikasi grup perangkat klien, dan menentukan kebijakan otorisasi perangkat klien yang menentukan izin untuk setiap grup perangkat.

Objek ini berisi informasi berikut:

`formatVersion`

Versi format untuk objek konfigurasi ini.

Pilih dari salah satu pilihan berikut:

- `2021-03-05`

`definitions`

grup perangkat untuk perangkat inti ini. Setiap definisi menentukan aturan pemilihan untuk mengevaluasi apakah perangkat klien adalah anggota grup. Setiap definisi juga

menentukan kebijakan izin yang akan diterapkan ke perangkat klien yang cocok dengan aturan pemilihan. Jika perangkat klien adalah anggota dari beberapa grup perangkat, izin perangkat terdiri dari kebijakan izin masing-masing grup.

Objek ini berisi informasi berikut:

groupNameKey

Nama grup perangkat ini. Ganti *groupNameKey*dengan nama yang membantu Anda mengidentifikasi grup perangkat ini.

Objek ini berisi informasi berikut:

`selectionRule`

Kueri yang menentukan perangkat klien mana yang menjadi anggota grup perangkat ini. Saat perangkat klien terhubung, perangkat inti mengevaluasi aturan pemilihan ini untuk menentukan apakah perangkat klien adalah anggota grup perangkat ini. Jika perangkat klien adalah anggota, perangkat inti akan menggunakan kebijakan grup perangkat ini untuk mengotorisasi tindakan perangkat klien.

Setiap aturan pemilihan terdiri dari setidaknya satu klausa aturan pemilihan, yang merupakan kueri ekspresi tunggal yang dapat mencocokkan perangkat klien. Aturan pemilihan menggunakan sintaks kueri yang sama dengan pengindeksan AWS IoT armada. Untuk informasi selengkapnya tentang sintaks aturan pemilihan, lihat sintaks kueri pengindeksan AWS IoT armada di Panduan Pengembang.AWS IoT Core

Gunakan wildcard * untuk mencocokkan beberapa perangkat klien dengan satu pilihan klausul aturan. Anda dapat menggunakan wildcard ini di akhir nama objek untuk mencocokkan perangkat klien yang namanya dimulai dengan string yang Anda tentukan. Anda juga dapat menggunakan wildcard ini untuk mencocokkan semua perangkat klien.

> **Note**
>
> Untuk memilih nilai yang berisi karakter titik dua (`:`), lepaskan titik dua dengan karakter garis miring terbalik (\\). Dalam format seperti JSON, Anda harus melepaskan karakter ris miring terbalik, sehingga Anda memasukkan dua karakter ris miring terbalik sebelum karakter titik dua. Sebagai contoh,

tentukan `thingName: MyTeam\\\\:ClientDevice1` untuk memilih objek yang namanya `MyTeam:ClientDevice1`.

Anda dapat menentukan sebagai berikut:

- `thingName` — Nama objek AWS IoT perangkat klien.

Example Contoh aturan pemilihan

Aturan seleksi berikut cocok dengan perangkat klien yang namanya `MyClientDevice1` atau `MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Contoh aturan seleksi (gunakan wildcard)

Aturan seleksi berikut cocok dengan perangkat klien yang namanya dimulai dengan `MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Contoh aturan pemilihan (cocok dengan semua perangkat)

Aturan seleksi berikut cocok dengan semua perangkat klien.

```
thingName: *
```

`policyName`

Kebijakan izin yang berlaku untuk perangkat klien dalam grup perangkat ini. Tentukan nama kebijakan yang Anda tetapkan di objek `policies`.

`policies`

Kebijakan otorisasi perangkat klien untuk perangkat klien yang terhubung ke perangkat inti. Setiap kebijakan otorisasi menentukan serangkaian tindakan dan sumber daya tempat perangkat klien dapat melakukan tindakan tersebut.

Objek ini berisi informasi berikut:

policyNameKey

Nama kebijakan otorisasi ini. Ganti *policyNameKey*dengan nama yang membantu Anda mengidentifikasi kebijakan otorisasi ini. Anda menggunakan nama kebijakan ini untuk menentukan kebijakan yang berlaku untuk grup perangkat.

Objek ini berisi informasi berikut:

statementNameKey

Nama pernyataan kebijakan ini. Ganti *statementNameKey*dengan nama yang membantu Anda mengidentifikasi pernyataan kebijakan ini.

Objek ini berisi informasi berikut:

`operations`

Daftar operasi untuk mengizinkan sumber daya dalam kebijakan ini.

Anda dapat menyertakan salah satu dari operasi berikut:

- `mqtt:connect` — Memberikan izin untuk terhubung ke perangkat inti. Perangkat klien harus memiliki izin ini untuk menyambung ke perangkat inti.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:clientId:`*`deviceClientId`* — Batasi akses berdasarkan ID klien yang digunakan perangkat klien untuk terhubung ke broker MQTT perangkat inti. Ganti *deviceClientId*dengan ID klien yang akan digunakan.
- `mqtt:publish` — Memberikan izin untuk mempublikasikan pesan MQTT ke topik.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topic:`*`mqttTopic`* — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *MqttTopic* dengan topik yang akan digunakan.

    Sumber daya ini tidak mendukung wildcard topik MQTT.
- `mqtt:subscribe` — Memberikan izin untuk berlangganan filter topik MQTT untuk menerima pesan.

  Operasi ini mendukung sumber daya berikut:

- `mqtt:topicfilter:`*`mqttTopicFilter`* — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`mqttTopicFilter`*dengan filter topik yang akan digunakan.

  Sumber daya ini mendukung wildcard topik MQTT + dan #. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

  Perangkat klien dapat berlangganan filter topik yang tepat yang Anda izinkan. Misalnya, jika Anda mengizinkan perangkat klien untuk berlangganan sumber daya `mqtt:topicfilter:client/+/status`, perangkat klien dapat berlangganan `client/+/status` tetapi bukan `client/client1/status`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua tindakan.

`resources`

Daftar operasi yang akan mengizinkan sumber daya dalam kebijakan ini. Tentukan sumber daya yang sesuai dengan operasi dalam kebijakan ini. Misalnya, Anda dapat menentukan daftar sumber daya topik MQTT (`mqtt:topic:`*`mqttTopic`*) dalam kebijakan yang menentukan operasi `mqtt:publish`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua sumber daya. Anda tidak dapat menggunakan wildcard * untuk mencocokkan pengidentifikasi sumber daya parsial. Misalnya, Anda dapat menentukan **`"resources": "*"`**, tetapi Anda tidak dapat menentukan **`"resources": "mqtt:clientId:*"`**.

`statementDescription`

(Opsional) Deskripsi untuk pernyataan kebijakan ini.

`certificates`

(Opsional) Opsi konfigurasi sertifikat untuk perangkat inti ini. Objek ini berisi informasi berikut:

`serverCertificateValiditySeconds`

(Opsional) Jumlah waktu (dalam detik) setelah sertifikat server MQTT lokal kedaluwarsa. Anda dapat mengonfigurasi opsi ini untuk menyesuaikan seberapa sering perangkat klien memutuskan sambungan dan menyambung kembali ke perangkat inti.

Komponen ini memutar sertifikat server MQTT lokal 24 jam sebelum kedaluwarsa. Broker MQTT, seperti komponen broker Moquette MQTT, menghasilkan sertifikat baru dan memulai ulang. Ketika ini terjadi, semua perangkat klien yang terhubung ke perangkat inti ini terputus. Perangkat klien dapat terhubung kembali ke perangkat inti setelah periode waktu yang singkat.

Default: `604800` (7 hari)

Nilai minimum: `172800` (2 hari)

Nilai maksimum: `864000` (10 hari)

`performance`

(Opsional) Opsi konfigurasi kinerja untuk perangkat inti ini. Objek ini berisi informasi berikut:

`maxActiveAuthTokens`

(Opsional) Jumlah maksimum token otorisasi perangkat klien aktif. Anda dapat meningkatkan jumlah ini untuk mengaktifkan lebih banyak perangkat klien untuk terhubung ke perangkat inti tunggal, tanpa mengautentikasi ulang mereka.

Default: `2500`

`cloudRequestQueueSize`

(Opsional) Jumlah maksimum AWS Cloud permintaan untuk mengantri sebelum komponen ini menolak permintaan.

Default: `100`

`maxConcurrentCloudRequests`

(Opsional) Jumlah maksimum permintaan bersamaan untuk dikirim ke. AWS Cloud Anda dapat meningkatkan angka ini untuk meningkatkan kinerja otentikasi pada perangkat inti tempat Anda menghubungkan sejumlah besar perangkat klien.

Default: 1

`certificateAuthority`

(Opsional) Opsi konfigurasi otoritas sertifikat untuk mengganti otoritas perantara perangkat inti dengan otoritas sertifikat perantara Anda sendiri.

> **Note**
>
> Jika Anda mengonfigurasi perangkat inti Greengrass Anda dengan otoritas sertifikat khusus (CA) dan menggunakan CA yang sama untuk menerbitkan sertifikat perangkat klien, Greengrass melewati pemeriksaan kebijakan otorisasi untuk operasi MQTT perangkat klien. Komponen autentikasi perangkat klien sepenuhnya mempercayai klien menggunakan sertifikat yang ditandatangani oleh CA yang dikonfigurasi untuk digunakan.
> Untuk membatasi perilaku ini saat menggunakan CA kustom, buat dan tandatangani perangkat klien menggunakan CA atau CA perantara yang berbeda, lalu sesuaikan `certificateChainUri` bidang `certificateUri` dan untuk menunjuk ke CA perantara yang benar.

Objek ini berisi informasi berikut.

CertificateURI

Lokasi sertifikat. Ini bisa berupa URI sistem file atau URI yang menunjuk ke sertifikat yang disimpan dalam modul keamanan perangkat keras.

`certificateChainUri`

Lokasi rantai sertifikat untuk perangkat inti CA. Ini harus menjadi rantai sertifikat lengkap kembali ke CA root Anda. Ini bisa berupa URI sistem file atau URI yang menunjuk ke rantai sertifikat yang disimpan dalam modul keamanan perangkat keras.

`privateKeyUri`

Lokasi kunci pribadi perangkat inti. Ini bisa berupa URI sistem file atau URI yang menunjuk ke kunci pribadi sertifikat yang disimpan dalam modul keamanan perangkat keras.

`security`

(Opsional) Opsi konfigurasi keamanan untuk perangkat inti ini. Objek ini berisi informasi berikut.

`clientDeviceTrustDurationMinutes`

Durasi dalam hitungan menit bahwa informasi otentikasi perangkat klien dapat dipercaya sebelum diperlukan untuk mengautentikasi ulang dengan perangkat inti. Nilai default adalah 1.

`metrics`

(Opsional) Opsi metrik untuk perangkat inti ini. Metrik kesalahan hanya akan ditampilkan jika ada kesalahan dengan autentikasi perangkat klien. Objek ini berisi informasi berikut:

`disableMetrics`

Jika `disableMetrics` bidang ditetapkan sebagai`true`, autentikasi perangkat klien tidak akan mengumpulkan metrik.

Default: `false`

`aggregatePeriodSeconds`

Periode agregasi dalam hitungan detik yang menentukan seberapa sering autentikasi perangkat klien mengumpulkan metrik dan mengirimkannya ke agen telemetri. Ini tidak mengubah seberapa sering metrik diterbitkan karena agen telemetri masih menerbitkannya sekali sehari.

Default: `3600`

startupTimeoutSeconds

(Opsional) Maksimum waktu dalam hitungan detik untuk memulai komponen. Status komponen berubah menjadi `BROKEN` jika melebihi batas waktu ini.

Default: `120`

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan yang ketat)

Contoh konfigurasi berikut menentukan untuk memungkinkan perangkat klien yang namanya dimulai dengan `MyClientDevice` untuk menyambung dan mempublikasikan/berlangganan semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
```

```
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
          ]
        }
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan permisif)

Contoh konfigurasi berikut menentukan untuk memungkinkan semua perangkat klien untuk terhubung dan mempublikasikan/berlangganan pada semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
```

```
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

v2.4.0 - v2.4.1

`deviceGroups`

Grup perangkat adalah grup perangkat klien yang memiliki izin untuk menyambung dan berkomunikasi dengan perangkat inti. Gunakan aturan pemilihan untuk mengidentifikasi grup perangkat klien, dan menentukan kebijakan otorisasi perangkat klien yang menentukan izin untuk setiap grup perangkat.

Objek ini berisi informasi berikut:

`formatVersion`

Versi format untuk objek konfigurasi ini.

Pilih dari salah satu pilihan berikut:

- `2021-03-05`

`definitions`

grup perangkat untuk perangkat inti ini. Setiap definisi menentukan aturan pemilihan untuk mengevaluasi apakah perangkat klien adalah anggota grup. Setiap definisi juga

menentukan kebijakan izin yang akan diterapkan ke perangkat klien yang cocok dengan aturan pemilihan. Jika perangkat klien adalah anggota dari beberapa grup perangkat, izin perangkat terdiri dari kebijakan izin masing-masing grup.

Objek ini berisi informasi berikut:

groupNameKey

Nama grup perangkat ini. Ganti *groupNameKey*dengan nama yang membantu Anda mengidentifikasi grup perangkat ini.

Objek ini berisi informasi berikut:

selectionRule

Kueri yang menentukan perangkat klien mana yang menjadi anggota grup perangkat ini. Saat perangkat klien terhubung, perangkat inti mengevaluasi aturan pemilihan ini untuk menentukan apakah perangkat klien adalah anggota grup perangkat ini. Jika perangkat klien adalah anggota, perangkat inti akan menggunakan kebijakan grup perangkat ini untuk mengotorisasi tindakan perangkat klien.

Setiap aturan pemilihan terdiri dari setidaknya satu klausa aturan pemilihan, yang merupakan kueri ekspresi tunggal yang dapat mencocokkan perangkat klien. Aturan pemilihan menggunakan sintaks kueri yang sama dengan pengindeksan AWS IoT armada. Untuk informasi selengkapnya tentang sintaks aturan pemilihan, lihat sintaks kueri pengindeksan AWS IoT armada di Panduan Pengembang.AWS IoT Core

Gunakan wildcard * untuk mencocokkan beberapa perangkat klien dengan satu pilihan klausul aturan. Anda dapat menggunakan wildcard ini di akhir nama objek untuk mencocokkan perangkat klien yang namanya dimulai dengan string yang Anda tentukan. Anda juga dapat menggunakan wildcard ini untuk mencocokkan semua perangkat klien.

> **Note**
>
> Untuk memilih nilai yang berisi karakter titik dua (`:`), lepaskan titik dua dengan karakter garis miring terbalik (\\). Dalam format seperti JSON, Anda harus melepaskan karakter ris miring terbalik, sehingga Anda memasukkan dua karakter ris miring terbalik sebelum karakter titik dua. Sebagai contoh,

tentukan `thingName: MyTeam\\\\:ClientDevice1` untuk memilih objek yang namanya `MyTeam:ClientDevice1`.

Anda dapat menentukan sebagai berikut:

- `thingName` — Nama objek AWS IoT perangkat klien.

Example Contoh aturan pemilihan

Aturan seleksi berikut cocok dengan perangkat klien yang namanya `MyClientDevice1` atau `MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Contoh aturan seleksi (gunakan wildcard)

Aturan seleksi berikut cocok dengan perangkat klien yang namanya dimulai dengan `MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Contoh aturan pemilihan (cocok dengan semua perangkat)

Aturan seleksi berikut cocok dengan semua perangkat klien.

```
thingName: *
```

`policyName`

Kebijakan izin yang berlaku untuk perangkat klien dalam grup perangkat ini. Tentukan nama kebijakan yang Anda tetapkan di objek `policies`.

`policies`

Kebijakan otorisasi perangkat klien untuk perangkat klien yang terhubung ke perangkat inti. Setiap kebijakan otorisasi menentukan serangkaian tindakan dan sumber daya tempat perangkat klien dapat melakukan tindakan tersebut.

Objek ini berisi informasi berikut:

policyNameKey

Nama kebijakan otorisasi ini. Ganti *policyNameKey*dengan nama yang membantu Anda mengidentifikasi kebijakan otorisasi ini. Anda menggunakan nama kebijakan ini untuk menentukan kebijakan yang berlaku untuk grup perangkat.

Objek ini berisi informasi berikut:

statementNameKey

Nama pernyataan kebijakan ini. Ganti *statementNameKey*dengan nama yang membantu Anda mengidentifikasi pernyataan kebijakan ini.

Objek ini berisi informasi berikut:

`operations`

Daftar operasi untuk mengizinkan sumber daya dalam kebijakan ini.

Anda dapat menyertakan salah satu dari operasi berikut:

- `mqtt:connect` — Memberikan izin untuk terhubung ke perangkat inti. Perangkat klien harus memiliki izin ini untuk menyambung ke perangkat inti.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:clientId:`*`deviceClientId`* — Batasi akses berdasarkan ID klien yang digunakan perangkat klien untuk terhubung ke broker MQTT perangkat inti. Ganti *deviceClientId*dengan ID klien yang akan digunakan.
- `mqtt:publish` — Memberikan izin untuk mempublikasikan pesan MQTT ke topik.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topic:`*`mqttTopic`* — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *MqttTopic* dengan topik yang akan digunakan.

    Sumber daya ini tidak mendukung wildcard topik MQTT.
- `mqtt:subscribe` — Memberikan izin untuk berlangganan filter topik MQTT untuk menerima pesan.

  Operasi ini mendukung sumber daya berikut:

- `mqtt:topicfilter:`*`mqttTopicFilter`* — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`mqttTopicFilter`*dengan filter topik yang akan digunakan.

  Sumber daya ini mendukung wildcard topik MQTT + dan #. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

  Perangkat klien dapat berlangganan filter topik yang tepat yang Anda izinkan. Misalnya, jika Anda mengizinkan perangkat klien untuk berlangganan sumber daya `mqtt:topicfilter:client/+/status`, perangkat klien dapat berlangganan `client/+/status` tetapi bukan `client/client1/status`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua tindakan.

`resources`

Daftar operasi yang akan mengizinkan sumber daya dalam kebijakan ini. Tentukan sumber daya yang sesuai dengan operasi dalam kebijakan ini. Misalnya, Anda dapat menentukan daftar sumber daya topik MQTT (`mqtt:topic:`*`mqttTopic`*) dalam kebijakan yang menentukan operasi `mqtt:publish`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua sumber daya. Anda tidak dapat menggunakan wildcard * untuk mencocokkan pengidentifikasi sumber daya parsial. Misalnya, Anda dapat menentukan **`"resources": "*"`**, tetapi Anda tidak dapat menentukan **`"resources": "mqtt:clientId:*"`**.

`statementDescription`

(Opsional) Deskripsi untuk pernyataan kebijakan ini.

`certificates`

(Opsional) Opsi konfigurasi sertifikat untuk perangkat inti ini. Objek ini berisi informasi berikut:

`serverCertificateValiditySeconds`

(Opsional) Jumlah waktu (dalam detik) setelah sertifikat server MQTT lokal kedaluwarsa. Anda dapat mengonfigurasi opsi ini untuk menyesuaikan seberapa sering perangkat klien memutuskan sambungan dan menyambung kembali ke perangkat inti.

Komponen ini memutar sertifikat server MQTT lokal 24 jam sebelum kedaluwarsa. Broker MQTT, seperti komponen broker Moquette MQTT, menghasilkan sertifikat baru dan memulai ulang. Ketika ini terjadi, semua perangkat klien yang terhubung ke perangkat inti ini terputus. Perangkat klien dapat terhubung kembali ke perangkat inti setelah periode waktu yang singkat.

Default: `604800` (7 hari)

Nilai minimum: `172800` (2 hari)

Nilai maksimum: `864000` (10 hari)

`performance`

(Opsional) Opsi konfigurasi kinerja untuk perangkat inti ini. Objek ini berisi informasi berikut:

`maxActiveAuthTokens`

(Opsional) Jumlah maksimum token otorisasi perangkat klien aktif. Anda dapat meningkatkan jumlah ini untuk mengaktifkan lebih banyak perangkat klien untuk terhubung ke perangkat inti tunggal, tanpa mengautentikasi ulang mereka.

Default: `2500`

`cloudRequestQueueSize`

(Opsional) Jumlah maksimum AWS Cloud permintaan untuk mengantri sebelum komponen ini menolak permintaan.

Default: `100`

`maxConcurrentCloudRequests`

(Opsional) Jumlah maksimum permintaan bersamaan untuk dikirim ke. AWS Cloud Anda dapat meningkatkan angka ini untuk meningkatkan kinerja otentikasi pada perangkat inti tempat Anda menghubungkan sejumlah besar perangkat klien.

Default: 1

`certificateAuthority`

(Opsional) Opsi konfigurasi otoritas sertifikat untuk mengganti otoritas perantara perangkat inti dengan otoritas sertifikat perantara Anda sendiri. Objek ini berisi informasi berikut.

Objek ini berisi informasi berikut:

CertificateURI

Lokasi sertifikat. Ini bisa berupa URI sistem file atau URI yang menunjuk ke sertifikat yang disimpan dalam modul keamanan perangkat keras.

`certificateChainUri`

Lokasi rantai sertifikat untuk perangkat inti CA. Ini harus menjadi rantai sertifikat lengkap kembali ke CA root Anda. Ini bisa berupa URI sistem file atau URI yang menunjuk ke rantai sertifikat yang disimpan dalam modul keamanan perangkat keras.

`privateKeyUri`

Lokasi kunci pribadi perangkat inti. Ini bisa berupa URI sistem file atau URI yang menunjuk ke kunci pribadi sertifikat yang disimpan dalam modul keamanan perangkat keras.

`security`

(Opsional) Opsi konfigurasi keamanan untuk perangkat inti ini. Objek ini berisi informasi berikut.

`clientDeviceTrustDurationMinutes`

Durasi dalam hitungan menit bahwa informasi otentikasi perangkat klien dapat dipercaya sebelum diperlukan untuk mengautentikasi ulang dengan perangkat inti. Nilai default adalah 1.

`metrics`

(Opsional) Opsi metrik untuk perangkat inti ini. Metrik kesalahan hanya akan ditampilkan jika ada kesalahan dengan autentikasi perangkat klien. Objek ini berisi informasi berikut:

`disableMetrics`

Jika `disableMetrics` bidang ditetapkan sebagai`true`, autentikasi perangkat klien tidak akan mengumpulkan metrik.

Default: `false`

`aggregatePeriodSeconds`

Periode agregasi dalam hitungan detik yang menentukan seberapa sering autentikasi perangkat klien mengumpulkan metrik dan mengirimkannya ke agen telemetri. Ini tidak

mengubah seberapa sering metrik diterbitkan karena agen telemetri masih menerbitkannya sekali sehari.

Default: `3600`

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan yang ketat)

Contoh konfigurasi berikut menentukan untuk memungkinkan perangkat klien yang namanya dimulai dengan `MyClientDevice` untuk menyambung dan mempublikasikan/berlangganan semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
```

```
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
          ]
        }
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan permisif)

Contoh konfigurasi berikut menentukan untuk memungkinkan semua perangkat klien untuk terhubung dan mempublikasikan/berlangganan pada semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

v2.3.x

`deviceGroups`

Grup perangkat adalah grup perangkat klien yang memiliki izin untuk menyambung dan berkomunikasi dengan perangkat inti. Gunakan aturan pemilihan untuk mengidentifikasi grup perangkat klien, dan menentukan kebijakan otorisasi perangkat klien yang menentukan izin untuk setiap grup perangkat.

Objek ini berisi informasi berikut:

`formatVersion`

Versi format untuk objek konfigurasi ini.

Pilih dari salah satu pilihan berikut:

- `2021-03-05`

`definitions`

grup perangkat untuk perangkat inti ini. Setiap definisi menentukan aturan pemilihan untuk mengevaluasi apakah perangkat klien adalah anggota grup. Setiap definisi juga menentukan kebijakan izin yang akan diterapkan ke perangkat klien yang cocok dengan aturan pemilihan. Jika perangkat klien adalah anggota dari beberapa grup perangkat, izin perangkat terdiri dari kebijakan izin masing-masing grup.

Objek ini berisi informasi berikut:

`groupNameKey`

Nama grup perangkat ini. Ganti *`groupNameKey`*dengan nama yang membantu Anda mengidentifikasi grup perangkat ini.

Objek ini berisi informasi berikut:

`selectionRule`

Kueri yang menentukan perangkat klien mana yang menjadi anggota grup perangkat ini. Saat perangkat klien terhubung, perangkat inti mengevaluasi aturan pemilihan ini untuk menentukan apakah perangkat klien adalah anggota grup perangkat ini. Jika perangkat klien adalah anggota, perangkat inti akan menggunakan kebijakan grup perangkat ini untuk mengotorisasi tindakan perangkat klien.

Setiap aturan pemilihan terdiri dari setidaknya satu klausa aturan pemilihan, yang merupakan kueri ekspresi tunggal yang dapat mencocokkan perangkat klien. Aturan pemilihan menggunakan sintaks kueri yang sama dengan pengindeksan AWS IoT armada. Untuk informasi selengkapnya tentang sintaks aturan pemilihan, lihat sintaks kueri pengindeksan AWS IoT armada di Panduan Pengembang.AWS IoT Core

Gunakan wildcard * untuk mencocokkan beberapa perangkat klien dengan satu pilihan klausul aturan. Anda dapat menggunakan wildcard ini di akhir nama objek untuk mencocokkan perangkat klien yang namanya dimulai dengan string yang Anda tentukan. Anda juga dapat menggunakan wildcard ini untuk mencocokkan semua perangkat klien.

> **Note**
>
> Untuk memilih nilai yang berisi karakter titik dua (`:`), lepaskan titik dua dengan karakter garis miring terbalik (\\). Dalam format seperti JSON, Anda harus melepaskan karakter ris miring terbalik, sehingga Anda memasukkan dua karakter ris miring terbalik sebelum karakter titik dua. Sebagai contoh, tentukan `thingName: MyTeam\\\\:ClientDevice1` untuk memilih objek yang namanya `MyTeam:ClientDevice1`.

Anda dapat menentukan sebagai berikut:

- `thingName` — Nama objek AWS IoT perangkat klien.

Example Contoh aturan pemilihan

Aturan seleksi berikut cocok dengan perangkat klien yang namanya `MyClientDevice1` atau `MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Contoh aturan seleksi (gunakan wildcard)

Aturan seleksi berikut cocok dengan perangkat klien yang namanya dimulai dengan `MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Contoh aturan pemilihan (cocok dengan semua perangkat)

Aturan seleksi berikut cocok dengan semua perangkat klien.

```
thingName: *
```

`policyName`

Kebijakan izin yang berlaku untuk perangkat klien dalam grup perangkat ini. Tentukan nama kebijakan yang Anda tetapkan di objek `policies`.

`policies`

Kebijakan otorisasi perangkat klien untuk perangkat klien yang terhubung ke perangkat inti. Setiap kebijakan otorisasi menentukan serangkaian tindakan dan sumber daya tempat perangkat klien dapat melakukan tindakan tersebut.

Objek ini berisi informasi berikut:

`policyNameKey`

Nama kebijakan otorisasi ini. Ganti *`policyNameKey`*dengan nama yang membantu Anda mengidentifikasi kebijakan otorisasi ini. Anda menggunakan nama kebijakan ini untuk menentukan kebijakan yang berlaku untuk grup perangkat.

Objek ini berisi informasi berikut:

`statementNameKey`

Nama pernyataan kebijakan ini. Ganti *`statementNameKey`*dengan nama yang membantu Anda mengidentifikasi pernyataan kebijakan ini.

Objek ini berisi informasi berikut:

`operations`

Daftar operasi untuk mengizinkan sumber daya dalam kebijakan ini.

Anda dapat menyertakan salah satu dari operasi berikut:

- `mqtt:connect` — Memberikan izin untuk terhubung ke perangkat inti. Perangkat klien harus memiliki izin ini untuk menyambung ke perangkat inti.

  Operasi ini mendukung sumber daya berikut:

- `mqtt:clientId:deviceClientId` — Batasi akses berdasarkan ID klien yang digunakan perangkat klien untuk terhubung ke broker MQTT perangkat inti. Ganti *`deviceClientId`*dengan ID klien yang akan digunakan.

- `mqtt:publish` — Memberikan izin untuk mempublikasikan pesan MQTT ke topik.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topic:mqttTopic` — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`MqttTopic`* dengan topik yang akan digunakan.

    Sumber daya ini tidak mendukung wildcard topik MQTT.

- `mqtt:subscribe` — Memberikan izin untuk berlangganan filter topik MQTT untuk menerima pesan.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topicfilter:mqttTopicFilter` — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`mqttTopicFilter`*dengan filter topik yang akan digunakan.

    Sumber daya ini mendukung wildcard topik MQTT + dan #. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

    Perangkat klien dapat berlangganan filter topik yang tepat yang Anda izinkan. Misalnya, jika Anda mengizinkan perangkat klien untuk berlangganan sumber daya `mqtt:topicfilter:client/+/status`, perangkat klien dapat berlangganan `client/+/status` tetapi bukan `client/client1/status`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua tindakan.

`resources`

Daftar operasi yang akan mengizinkan sumber daya dalam kebijakan ini. Tentukan sumber daya yang sesuai dengan operasi dalam kebijakan ini. Misalnya, Anda dapat menentukan daftar sumber daya topik MQTT (`mqtt:topic:mqttTopic`) dalam kebijakan yang menentukan operasi `mqtt:publish`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua sumber daya. Anda tidak dapat menggunakan wildcard * untuk mencocokkan pengidentifikasi sumber daya parsial. Misalnya, Anda dapat menentukan **`"resources": "*"`**, tetapi Anda tidak dapat menentukan **`"resources": "mqtt:clientId:*"`**.

`statementDescription`

(Opsional) Deskripsi untuk pernyataan kebijakan ini.

`certificates`

(Opsional) Opsi konfigurasi sertifikat untuk perangkat inti ini. Objek ini berisi informasi berikut:

`serverCertificateValiditySeconds`

(Opsional) Jumlah waktu (dalam detik) setelah sertifikat server MQTT lokal kedaluwarsa. Anda dapat mengonfigurasi opsi ini untuk menyesuaikan seberapa sering perangkat klien memutuskan sambungan dan menyambung kembali ke perangkat inti.

Komponen ini memutar sertifikat server MQTT lokal 24 jam sebelum kedaluwarsa. Broker MQTT, seperti komponen broker Moquette MQTT, menghasilkan sertifikat baru dan memulai ulang. Ketika ini terjadi, semua perangkat klien yang terhubung ke perangkat inti ini terputus. Perangkat klien dapat terhubung kembali ke perangkat inti setelah periode waktu yang singkat.

Default: `604800` (7 hari)

Nilai minimum: `172800` (2 hari)

Nilai maksimum: `864000` (10 hari)

`performance`

(Opsional) Opsi konfigurasi kinerja untuk perangkat inti ini. Objek ini berisi informasi berikut:

`maxActiveAuthTokens`

(Opsional) Jumlah maksimum token otorisasi perangkat klien aktif. Anda dapat meningkatkan jumlah ini untuk mengaktifkan lebih banyak perangkat klien untuk terhubung ke perangkat inti tunggal tanpa mengautentikasi ulang mereka.

Default: `2500`

`cloudRequestQueueSize`

(Opsional) Jumlah maksimum AWS Cloud permintaan untuk mengantri sebelum komponen ini menolak permintaan.

Default: `100`

`maxConcurrentCloudRequests`

(Opsional) Jumlah maksimum permintaan bersamaan untuk dikirim ke. AWS Cloud Anda dapat meningkatkan angka ini untuk meningkatkan kinerja otentikasi pada perangkat inti tempat Anda menghubungkan sejumlah besar perangkat klien.

Default: 1

`certificateAuthority`

(Opsional) Opsi konfigurasi otoritas sertifikat untuk mengganti otoritas perantara perangkat inti dengan otoritas sertifikat perantara Anda sendiri. Objek ini berisi informasi berikut.

CertificateURI

Lokasi sertifikat. Ini bisa berupa URI sistem file atau URI yang menunjuk ke sertifikat yang disimpan dalam modul keamanan perangkat keras.

`certificateChainUri`

Lokasi rantai sertifikat untuk perangkat inti CA. Ini harus menjadi rantai sertifikat lengkap kembali ke CA root Anda. Ini bisa berupa URI sistem file atau URI yang menunjuk ke rantai sertifikat yang disimpan dalam modul keamanan perangkat keras.

`privateKeyUri`

Lokasi kunci pribadi perangkat inti. Ini bisa berupa URI sistem file atau URI yang menunjuk ke kunci pribadi sertifikat yang disimpan dalam modul keamanan perangkat keras.

`security`

(Opsional) Opsi konfigurasi keamanan untuk perangkat inti ini. Objek ini berisi informasi berikut.

`clientDeviceTrustDurationMinutes`

Durasi dalam hitungan menit bahwa informasi otentikasi perangkat klien dapat dipercaya sebelum diperlukan untuk mengautentikasi ulang dengan perangkat inti. Nilai default adalah 1.

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan yang ketat)

Contoh konfigurasi berikut menentukan untuk memungkinkan perangkat klien yang namanya dimulai dengan `MyClientDevice` untuk menyambung dan mempublikasikan/berlangganan semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
```

```
          ]
        }
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan permisif)

Contoh konfigurasi berikut menentukan untuk memungkinkan semua perangkat klien untuk terhubung dan mempublikasikan/berlangganan pada semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

v2.2.x

`deviceGroups`

Grup perangkat adalah grup perangkat klien yang memiliki izin untuk menyambung dan berkomunikasi dengan perangkat inti. Gunakan aturan pemilihan untuk mengidentifikasi grup

perangkat klien, dan menentukan kebijakan otorisasi perangkat klien yang menentukan izin untuk setiap grup perangkat.

Objek ini berisi informasi berikut:

`formatVersion`

Versi format untuk objek konfigurasi ini.

Pilih dari salah satu pilihan berikut:

- `2021-03-05`

`definitions`

grup perangkat untuk perangkat inti ini. Setiap definisi menentukan aturan pemilihan untuk mengevaluasi apakah perangkat klien adalah anggota grup. Setiap definisi juga menentukan kebijakan izin yang akan diterapkan ke perangkat klien yang cocok dengan aturan pemilihan. Jika perangkat klien adalah anggota dari beberapa grup perangkat, izin perangkat terdiri dari kebijakan izin masing-masing grup.

Objek ini berisi informasi berikut:

`groupNameKey`

Nama grup perangkat ini. Ganti *`groupNameKey`*dengan nama yang membantu Anda mengidentifikasi grup perangkat ini.

Objek ini berisi informasi berikut:

`selectionRule`

Kueri yang menentukan perangkat klien mana yang menjadi anggota grup perangkat ini. Saat perangkat klien terhubung, perangkat inti mengevaluasi aturan pemilihan ini untuk menentukan apakah perangkat klien adalah anggota grup perangkat ini. Jika perangkat klien adalah anggota, perangkat inti akan menggunakan kebijakan grup perangkat ini untuk mengotorisasi tindakan perangkat klien.

Setiap aturan pemilihan terdiri dari setidaknya satu klausa aturan pemilihan, yang merupakan kueri ekspresi tunggal yang dapat mencocokkan perangkat klien. Aturan pemilihan menggunakan sintaks kueri yang sama dengan pengindeksan AWS IoT armada. Untuk informasi selengkapnya tentang sintaks aturan pemilihan, lihat

sintaks kueri pengindeksan AWS IoT armada di Panduan Pengembang.AWS IoT Core

Gunakan wildcard * untuk mencocokkan beberapa perangkat klien dengan satu pilihan klausul aturan. Anda dapat menggunakan wildcard ini di akhir nama objek untuk mencocokkan perangkat klien yang namanya dimulai dengan string yang Anda tentukan. Anda juga dapat menggunakan wildcard ini untuk mencocokkan semua perangkat klien.

> **Note**
>
> Untuk memilih nilai yang berisi karakter titik dua (`:`), lepaskan titik dua dengan karakter garis miring terbalik (\\). Dalam format seperti JSON, Anda harus melepaskan karakter ris miring terbalik, sehingga Anda memasukkan dua karakter ris miring terbalik sebelum karakter titik dua. Sebagai contoh, tentukan `thingName: MyTeam\\\\:ClientDevice1` untuk memilih objek yang namanya `MyTeam:ClientDevice1`.

Anda dapat menentukan sebagai berikut:

- `thingName` — Nama objek AWS IoT perangkat klien.

Example Contoh aturan pemilihan

Aturan seleksi berikut cocok dengan perangkat klien yang namanya `MyClientDevice1` atau `MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Contoh aturan seleksi (gunakan wildcard)

Aturan seleksi berikut cocok dengan perangkat klien yang namanya dimulai dengan `MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Contoh aturan pemilihan (cocok dengan semua perangkat)

Aturan seleksi berikut cocok dengan semua perangkat klien.

```
thingName: *
```

`policyName`

Kebijakan izin yang berlaku untuk perangkat klien dalam grup perangkat ini. Tentukan nama kebijakan yang Anda tetapkan di objek `policies`.

`policies`

Kebijakan otorisasi perangkat klien untuk perangkat klien yang terhubung ke perangkat inti. Setiap kebijakan otorisasi menentukan serangkaian tindakan dan sumber daya tempat perangkat klien dapat melakukan tindakan tersebut.

Objek ini berisi informasi berikut:

`policyNameKey`

Nama kebijakan otorisasi ini. Ganti *`policyNameKey`*dengan nama yang membantu Anda mengidentifikasi kebijakan otorisasi ini. Anda menggunakan nama kebijakan ini untuk menentukan kebijakan yang berlaku untuk grup perangkat.

Objek ini berisi informasi berikut:

`statementNameKey`

Nama pernyataan kebijakan ini. Ganti *`statementNameKey`*dengan nama yang membantu Anda mengidentifikasi pernyataan kebijakan ini.

Objek ini berisi informasi berikut:

`operations`

Daftar operasi untuk mengizinkan sumber daya dalam kebijakan ini.

Anda dapat menyertakan salah satu dari operasi berikut:

- `mqtt:connect` — Memberikan izin untuk terhubung ke perangkat inti. Perangkat klien harus memiliki izin ini untuk menyambung ke perangkat inti.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:clientId:`*`deviceClientId`* — Batasi akses berdasarkan ID klien yang digunakan perangkat klien untuk terhubung ke broker MQTT perangkat inti. Ganti *`deviceClientId`*dengan ID klien yang akan digunakan.

- `mqtt:publish` — Memberikan izin untuk mempublikasikan pesan MQTT ke topik.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topic:mqttTopic` — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`MqttTopic`* dengan topik yang akan digunakan.

    Sumber daya ini tidak mendukung wildcard topik MQTT.

- `mqtt:subscribe` — Memberikan izin untuk berlangganan filter topik MQTT untuk menerima pesan.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topicfilter:mqttTopicFilter` — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`mqttTopicFilter`*dengan filter topik yang akan digunakan.

    Sumber daya ini mendukung wildcard topik MQTT + dan #. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

    Perangkat klien dapat berlangganan filter topik yang tepat yang Anda izinkan. Misalnya, jika Anda mengizinkan perangkat klien untuk berlangganan sumber daya `mqtt:topicfilter:client/+/status`, perangkat klien dapat berlangganan `client/+/status` tetapi bukan `client/client1/status`.

  Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua tindakan.

`resources`

Daftar operasi yang akan mengizinkan sumber daya dalam kebijakan ini. Tentukan sumber daya yang sesuai dengan operasi dalam kebijakan ini. Misalnya, Anda dapat menentukan daftar sumber daya topik MQTT (`mqtt:topic:`*`mqttTopic`*) dalam kebijakan yang menentukan operasi `mqtt:publish`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua sumber daya. Anda tidak dapat menggunakan wildcard * untuk mencocokkan pengidentifikasi sumber daya parsial. Misalnya, Anda dapat menentukan

"resources": "*", tetapi Anda tidak dapat menentukan **"resources": "mqtt:clientId:*"**.

`statementDescription`

(Opsional) Deskripsi untuk pernyataan kebijakan ini.

`certificates`

(Opsional) Opsi konfigurasi sertifikat untuk perangkat inti ini. Objek ini berisi informasi berikut:

`serverCertificateValiditySeconds`

(Opsional) Jumlah waktu (dalam detik) setelah sertifikat server MQTT lokal kedaluwarsa. Anda dapat mengonfigurasi opsi ini untuk menyesuaikan seberapa sering perangkat klien memutuskan sambungan dan menyambung kembali ke perangkat inti.

Komponen ini memutar sertifikat server MQTT lokal 24 jam sebelum kedaluwarsa. Broker MQTT, seperti komponen broker Moquette MQTT, menghasilkan sertifikat baru dan memulai ulang. Ketika ini terjadi, semua perangkat klien yang terhubung ke perangkat inti ini terputus. Perangkat klien dapat terhubung kembali ke perangkat inti setelah periode waktu yang singkat.

Default: `604800` (7 hari)

Nilai minimum: `172800` (2 hari)

Nilai maksimum: `864000` (10 hari)

`performance`

(Opsional) Opsi konfigurasi kinerja untuk perangkat inti ini. Objek ini berisi informasi berikut:

`maxActiveAuthTokens`

(Opsional) Jumlah maksimum token otorisasi perangkat klien aktif. Anda dapat meningkatkan jumlah ini untuk mengaktifkan lebih banyak perangkat klien untuk terhubung ke perangkat inti tunggal tanpa mengautentikasi ulang mereka.

Default: `2500`

`cloudRequestQueueSize`

(Opsional) Jumlah maksimum AWS Cloud permintaan untuk mengantri sebelum komponen ini menolak permintaan.

Default: `100`

`maxConcurrentCloudRequests`

(Opsional) Jumlah maksimum permintaan bersamaan untuk dikirim ke. AWS Cloud Anda dapat meningkatkan angka ini untuk meningkatkan kinerja otentikasi pada perangkat inti tempat Anda menghubungkan sejumlah besar perangkat klien.

Default: 1

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan yang ketat)

Contoh konfigurasi berikut menentukan untuk memungkinkan perangkat klien yang namanya dimulai dengan `MyClientDevice` untuk menyambung dan mempublikasikan/berlangganan semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
```

```
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
          ]
        }
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan permisif)

Contoh konfigurasi berikut menentukan untuk memungkinkan semua perangkat klien untuk terhubung dan mempublikasikan/berlangganan pada semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
```

```
    }
  }
}
```

v2.1.x

`deviceGroups`

Grup perangkat adalah grup perangkat klien yang memiliki izin untuk menyambung dan berkomunikasi dengan perangkat inti. Gunakan aturan pemilihan untuk mengidentifikasi grup perangkat klien, dan menentukan kebijakan otorisasi perangkat klien yang menentukan izin untuk setiap grup perangkat.

Objek ini berisi informasi berikut:

`formatVersion`

Versi format untuk objek konfigurasi ini.

Pilih dari salah satu pilihan berikut:

- `2021-03-05`

`definitions`

grup perangkat untuk perangkat inti ini. Setiap definisi menentukan aturan pemilihan untuk mengevaluasi apakah perangkat klien adalah anggota grup. Setiap definisi juga menentukan kebijakan izin yang akan diterapkan ke perangkat klien yang cocok dengan aturan pemilihan. Jika perangkat klien adalah anggota dari beberapa grup perangkat, izin perangkat terdiri dari kebijakan izin masing-masing grup.

Objek ini berisi informasi berikut:

`groupNameKey`

Nama grup perangkat ini. Ganti *`groupNameKey`*dengan nama yang membantu Anda mengidentifikasi grup perangkat ini.

Objek ini berisi informasi berikut:

`selectionRule`

Kueri yang menentukan perangkat klien mana yang menjadi anggota grup perangkat ini. Saat perangkat klien terhubung, perangkat inti mengevaluasi

aturan pemilihan ini untuk menentukan apakah perangkat klien adalah anggota grup perangkat ini. Jika perangkat klien adalah anggota, perangkat inti akan menggunakan kebijakan grup perangkat ini untuk mengotorisasi tindakan perangkat klien.

Setiap aturan pemilihan terdiri dari setidaknya satu klausa aturan pemilihan, yang merupakan kueri ekspresi tunggal yang dapat mencocokkan perangkat klien. Aturan pemilihan menggunakan sintaks kueri yang sama dengan pengindeksan AWS IoT armada. Untuk informasi selengkapnya tentang sintaks aturan pemilihan, lihat sintaks kueri pengindeksan AWS IoT armada di Panduan Pengembang.AWS IoT Core

Gunakan wildcard * untuk mencocokkan beberapa perangkat klien dengan satu pilihan klausul aturan. Anda dapat menggunakan wildcard ini di akhir nama objek untuk mencocokkan perangkat klien yang namanya dimulai dengan string yang Anda tentukan. Anda juga dapat menggunakan wildcard ini untuk mencocokkan semua perangkat klien.

> **Note**
>
> Untuk memilih nilai yang berisi karakter titik dua (`:`), lepaskan titik dua dengan karakter garis miring terbalik (\\). Dalam format seperti JSON, Anda harus melepaskan karakter ris miring terbalik, sehingga Anda memasukkan dua karakter ris miring terbalik sebelum karakter titik dua. Sebagai contoh, tentukan `thingName: MyTeam\\\\:ClientDevice1` untuk memilih objek yang namanya `MyTeam:ClientDevice1`.

Anda dapat menentukan sebagai berikut:

- `thingName` — Nama objek AWS IoT perangkat klien.

Example Contoh aturan pemilihan

Aturan seleksi berikut cocok dengan perangkat klien yang namanya `MyClientDevice1` atau `MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Contoh aturan seleksi (gunakan wildcard)

Aturan seleksi berikut cocok dengan perangkat klien yang namanya dimulai dengan `MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Contoh aturan pemilihan (cocok dengan semua perangkat)

Aturan seleksi berikut cocok dengan semua perangkat klien.

```
thingName: *
```

`policyName`

Kebijakan izin yang berlaku untuk perangkat klien dalam grup perangkat ini. Tentukan nama kebijakan yang Anda tetapkan di objek `policies`.

`policies`

Kebijakan otorisasi perangkat klien untuk perangkat klien yang terhubung ke perangkat inti. Setiap kebijakan otorisasi menentukan serangkaian tindakan dan sumber daya tempat perangkat klien dapat melakukan tindakan tersebut.

Objek ini berisi informasi berikut:

`policyNameKey`

Nama kebijakan otorisasi ini. Ganti *`policyNameKey`*dengan nama yang membantu Anda mengidentifikasi kebijakan otorisasi ini. Anda menggunakan nama kebijakan ini untuk menentukan kebijakan yang berlaku untuk grup perangkat.

Objek ini berisi informasi berikut:

`statementNameKey`

Nama pernyataan kebijakan ini. Ganti *`statementNameKey`*dengan nama yang membantu Anda mengidentifikasi pernyataan kebijakan ini.

Objek ini berisi informasi berikut:

`operations`

Daftar operasi untuk mengizinkan sumber daya dalam kebijakan ini.

Anda dapat menyertakan salah satu dari operasi berikut:

- `mqtt:connect` — Memberikan izin untuk terhubung ke perangkat inti. Perangkat klien harus memiliki izin ini untuk menyambung ke perangkat inti.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:clientId:`*`deviceClientId`* — Batasi akses berdasarkan ID klien yang digunakan perangkat klien untuk terhubung ke broker MQTT perangkat inti. Ganti *`deviceClientId`*dengan ID klien yang akan digunakan.

- `mqtt:publish` — Memberikan izin untuk mempublikasikan pesan MQTT ke topik.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topic:`*`mqttTopic`* — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`MqttTopic`* dengan topik yang akan digunakan.

    Sumber daya ini tidak mendukung wildcard topik MQTT.

- `mqtt:subscribe` — Memberikan izin untuk berlangganan filter topik MQTT untuk menerima pesan.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topicfilter:`*`mqttTopicFilter`* — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`mqttTopicFilter`*dengan filter topik yang akan digunakan.

    Sumber daya ini mendukung wildcard topik MQTT + dan #. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

    Perangkat klien dapat berlangganan filter topik yang tepat yang Anda izinkan. Misalnya, jika Anda mengizinkan perangkat klien untuk berlangganan sumber daya `mqtt:topicfilter:client/+/status`, perangkat klien dapat berlangganan `client/+/status` tetapi bukan `client/client1/status`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua tindakan.

`resources`

Daftar operasi yang akan mengizinkan sumber daya dalam kebijakan ini. Tentukan sumber daya yang sesuai dengan operasi dalam kebijakan ini. Misalnya, Anda dapat menentukan daftar sumber daya topik MQTT (`mqtt:topic:`*`mqttTopic`*) dalam kebijakan yang menentukan operasi `mqtt:publish`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua sumber daya. Anda tidak dapat menggunakan wildcard * untuk mencocokkan pengidentifikasi sumber daya parsial. Misalnya, Anda dapat menentukan **`"resources": "*"`**, tetapi Anda tidak dapat menentukan **`"resources": "mqtt:clientId:*"`**.

`statementDescription`

(Opsional) Deskripsi untuk pernyataan kebijakan ini.

`certificates`

(Opsional) Opsi konfigurasi sertifikat untuk perangkat inti ini. Objek ini berisi informasi berikut:

`serverCertificateValiditySeconds`

(Opsional) Jumlah waktu (dalam detik) setelah sertifikat server MQTT lokal kedaluwarsa. Anda dapat mengonfigurasi opsi ini untuk menyesuaikan seberapa sering perangkat klien memutuskan sambungan dan menyambung kembali ke perangkat inti.

Komponen ini memutar sertifikat server MQTT lokal 24 jam sebelum kedaluwarsa. Broker MQTT, seperti komponen broker Moquette MQTT, menghasilkan sertifikat baru dan memulai ulang. Ketika ini terjadi, semua perangkat klien yang terhubung ke perangkat inti ini terputus. Perangkat klien dapat terhubung kembali ke perangkat inti setelah periode waktu yang singkat.

Default: `604800` (7 hari)

Nilai minimum: `172800` (2 hari)

Nilai maksimum: `864000` (10 hari)

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan yang ketat)

Contoh konfigurasi berikut menentukan untuk memungkinkan perangkat klien yang namanya dimulai dengan `MyClientDevice` untuk menyambung dan mempublikasikan/berlangganan semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
```

```
          ]
        }
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan permisif)

Contoh konfigurasi berikut menentukan untuk memungkinkan semua perangkat klien untuk terhubung dan mempublikasikan/berlangganan pada semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

v2.0.x

`deviceGroups`

Grup perangkat adalah grup perangkat klien yang memiliki izin untuk menyambung dan berkomunikasi dengan perangkat inti. Gunakan aturan pemilihan untuk mengidentifikasi grup

perangkat klien, dan menentukan kebijakan otorisasi perangkat klien yang menentukan izin untuk setiap grup perangkat.

Objek ini berisi informasi berikut:

`formatVersion`

Versi format untuk objek konfigurasi ini.

Pilih dari salah satu pilihan berikut:

- `2021-03-05`

`definitions`

grup perangkat untuk perangkat inti ini. Setiap definisi menentukan aturan pemilihan untuk mengevaluasi apakah perangkat klien adalah anggota grup. Setiap definisi juga menentukan kebijakan izin yang akan diterapkan ke perangkat klien yang cocok dengan aturan pemilihan. Jika perangkat klien adalah anggota dari beberapa grup perangkat, izin perangkat terdiri dari kebijakan izin masing-masing grup.

Objek ini berisi informasi berikut:

`groupNameKey`

Nama grup perangkat ini. Ganti *`groupNameKey`*dengan nama yang membantu Anda mengidentifikasi grup perangkat ini.

Objek ini berisi informasi berikut:

`selectionRule`

Kueri yang menentukan perangkat klien mana yang menjadi anggota grup perangkat ini. Saat perangkat klien terhubung, perangkat inti mengevaluasi aturan pemilihan ini untuk menentukan apakah perangkat klien adalah anggota grup perangkat ini. Jika perangkat klien adalah anggota, perangkat inti akan menggunakan kebijakan grup perangkat ini untuk mengotorisasi tindakan perangkat klien.

Setiap aturan pemilihan terdiri dari setidaknya satu klausa aturan pemilihan, yang merupakan kueri ekspresi tunggal yang dapat mencocokkan perangkat klien. Aturan pemilihan menggunakan sintaks kueri yang sama dengan pengindeksan AWS IoT armada. Untuk informasi selengkapnya tentang sintaks aturan pemilihan, lihat

sintaks kueri pengindeksan AWS IoT armada di Panduan Pengembang.AWS IoT Core

Gunakan wildcard * untuk mencocokkan beberapa perangkat klien dengan satu pilihan klausul aturan. Anda dapat menggunakan wildcard ini di akhir nama objek untuk mencocokkan perangkat klien yang namanya dimulai dengan string yang Anda tentukan. Anda juga dapat menggunakan wildcard ini untuk mencocokkan semua perangkat klien.

> **Note**
>
> Untuk memilih nilai yang berisi karakter titik dua (`:`), lepaskan titik dua dengan karakter garis miring terbalik (\\). Dalam format seperti JSON, Anda harus melepaskan karakter ris miring terbalik, sehingga Anda memasukkan dua karakter ris miring terbalik sebelum karakter titik dua. Sebagai contoh, tentukan `thingName: MyTeam\\\\:ClientDevice1` untuk memilih objek yang namanya `MyTeam:ClientDevice1`.

Anda dapat menentukan sebagai berikut:

- `thingName` — Nama objek AWS IoT perangkat klien.

Example Contoh aturan pemilihan

Aturan seleksi berikut cocok dengan perangkat klien yang namanya `MyClientDevice1` atau `MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Contoh aturan seleksi (gunakan wildcard)

Aturan seleksi berikut cocok dengan perangkat klien yang namanya dimulai dengan `MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Contoh aturan pemilihan (cocok dengan semua perangkat)

Aturan seleksi berikut cocok dengan semua perangkat klien.

```
thingName: *
```

`policyName`

Kebijakan izin yang berlaku untuk perangkat klien dalam grup perangkat ini. Tentukan nama kebijakan yang Anda tetapkan di objek `policies`.

`policies`

Kebijakan otorisasi perangkat klien untuk perangkat klien yang terhubung ke perangkat inti. Setiap kebijakan otorisasi menentukan serangkaian tindakan dan sumber daya tempat perangkat klien dapat melakukan tindakan tersebut.

Objek ini berisi informasi berikut:

`policyNameKey`

Nama kebijakan otorisasi ini. Ganti *`policyNameKey`*dengan nama yang membantu Anda mengidentifikasi kebijakan otorisasi ini. Anda menggunakan nama kebijakan ini untuk menentukan kebijakan yang berlaku untuk grup perangkat.

Objek ini berisi informasi berikut:

`statementNameKey`

Nama pernyataan kebijakan ini. Ganti *`statementNameKey`*dengan nama yang membantu Anda mengidentifikasi pernyataan kebijakan ini.

Objek ini berisi informasi berikut:

`operations`

Daftar operasi untuk mengizinkan sumber daya dalam kebijakan ini.

Anda dapat menyertakan salah satu dari operasi berikut:

- `mqtt:connect` — Memberikan izin untuk terhubung ke perangkat inti. Perangkat klien harus memiliki izin ini untuk menyambung ke perangkat inti.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:clientId:`*`deviceClientId`* — Batasi akses berdasarkan ID klien yang digunakan perangkat klien untuk terhubung ke broker MQTT perangkat inti. Ganti *`deviceClientId`*dengan ID klien yang akan digunakan.

- `mqtt:publish` — Memberikan izin untuk mempublikasikan pesan MQTT ke topik.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topic:mqttTopic` — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`MqttTopic`* dengan topik yang akan digunakan.

    Sumber daya ini tidak mendukung wildcard topik MQTT.

- `mqtt:subscribe` — Memberikan izin untuk berlangganan filter topik MQTT untuk menerima pesan.

  Operasi ini mendukung sumber daya berikut:

  - `mqtt:topicfilter:mqttTopicFilter` — Membatasi akses berdasarkan topik MQTT di mana perangkat klien menerbitkan pesan. Ganti *`mqttTopicFilter`*dengan filter topik yang akan digunakan.

    Sumber daya ini mendukung wildcard topik MQTT + dan #. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

    Perangkat klien dapat berlangganan filter topik yang tepat yang Anda izinkan. Misalnya, jika Anda mengizinkan perangkat klien untuk berlangganan sumber daya `mqtt:topicfilter:client/+/status`, perangkat klien dapat berlangganan `client/+/status` tetapi bukan `client/client1/status`.

  Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua tindakan.

`resources`

Daftar operasi yang akan mengizinkan sumber daya dalam kebijakan ini. Tentukan sumber daya yang sesuai dengan operasi dalam kebijakan ini. Misalnya, Anda dapat menentukan daftar sumber daya topik MQTT (`mqtt:topic:`*`mqttTopic`*) dalam kebijakan yang menentukan operasi `mqtt:publish`.

Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua sumber daya. Anda tidak dapat menggunakan wildcard * untuk mencocokkan pengidentifikasi sumber daya parsial. Misalnya, Anda dapat menentukan

`"resources": "*"`, tetapi Anda tidak dapat menentukan **`"resources": "mqtt:clientId:*"`**.

`statementDescription`

(Opsional) Deskripsi untuk pernyataan kebijakan ini.

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan yang ketat)

Contoh konfigurasi berikut menentukan untuk memungkinkan perangkat klien yang namanya dimulai dengan `MyClientDevice` untuk menyambung dan mempublikasikan/berlangganan semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
```

```
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
          ]
        }
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (menggunakan kebijakan permisif)

Contoh konfigurasi berikut menentukan untuk memungkinkan semua perangkat klien untuk terhubung dan mempublikasikan/berlangganan pada semua topik.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

File log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.5.0	Fitur baru • Memungkinkan substitusi `${iot:Connection.Thing.ThingName}` variabel untuk sumber daya kebijakan. • Mengizinkan sumber daya kebijakan dengan wildcard seperti `mqtt:topic:my*` .

Versi	Perubahan
2.4.5	Fitur baru Menambahkan dukungan untuk awalan wildcard untuk memilih nama benda dengan parameter. `selectionRule` Perbaikan bug dan peningkatan Memperbaiki masalah saat sertifikat tidak diperbarui dengan informasi konektivitas baru dalam kasus tertentu.
2.4.4	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.4.3	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.4.2	Fitur baru Menambahkan opsi `startupTimeoutSeconds` konfigurasi baru.
2.4.1	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.4.0	Fitur baru • Menambahkan dukungan untuk autentikasi perangkat klien untuk memancarkan metrik operasional yang akan diterbitkan oleh agen telemetri. Perbaikan bug dan peningkatan • Memperbaiki masalah saat autentikasi perangkat klien membutuhkan waktu lebih dari 10 detik untuk memverifikasi identitas perangkat klien. • Peningkatan dan perbaikan kecil tambahan.
2.3.2	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk caching informasi nama host sehingga komponen menghasilkan subjek sertifikat dengan benar saat di-restart saat offline.
2.3.1	Perbaikan bug dan peningkatan • Memperbaiki kebocoran memori.

Versi	Perubahan
2.3.0	**Warning** Versi ini tidak lagi tersedia. Perbaikan dalam versi ini tersedia di versi yang lebih baru dari komponen ini. Fitur baru • Menambahkan dukungan untuk otentikasi offline perangkat klien sehingga mereka dapat terus terhubung ke perangkat inti ketika perangkat inti tidak terhubung ke Internet. • Menambahkan dukungan untuk otoritas sertifikat yang disediakan pelanggan yang digunakan perangkat inti sebagai sertifikat root untuk menghasilkan sertifikat broker MQTT.
2.2.3	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.2.2	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana sertifikat server MQTT lokal berputar lebih sering daripada yang dimaksudkan dalam skenario tertentu.
2.2.1	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.2.0	Fitur baru • Menambahkan dukungan untuk komponen kustom untuk memanggil operasi komunikasi antarproses (IPC) untuk mengautentikasi dan mengotorisasi perangkat klien. Anda dapat menggunakan operasi ini dalam komponen broker MQTT khusus, misalnya. Untuk informasi selengkapnya, lihat IPC: Mengautentikasi dan mengotorisasi perangkat klien. • Menambahkan `maxActiveAuthTokens` , `cloudQueueSize` , dan `threadPoolSize` opsi yang dapat Anda konfigurasikan untuk menyetel kinerja komponen ini.

Versi	Perubahan
2.1.0	Fitur baru • Menambahkan `serverCertificateValiditySeconds` opsi yang dapat Anda konfigurasi untuk menyesuaikan ketika sertifikat server broker MQTT kedaluwarsa. Anda dapat mengonfigurasi sertifikat server untuk kedaluwarsa setelah 2 hingga 10 hari. Perbaikan bug dan peningkatan • Memperbaiki masalah dengan cara komponen ini menangani pembaruan pengaturan ulang konfigurasi. • Memperbaiki masalah di mana sertifikat server MQTT lokal berputar lebih sering daripada yang dimaksudkan dalam skenario tertentu. Untuk menerapkan perbaikan ini, Anda juga harus menggunakan v2.1.0 atau yang lebih baru dari komponen broker Moquette MQTT. • Meningkatkan pesan yang dicatat oleh komponen ini saat memutar sertifikat. • Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.0.4	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.3	Perbaikan bug dan peningkatan • Kredenal sekarang disegarkan jika Anda memutar kunci pribadi perangkat inti. • Pembaruan untuk membuat pesan log lebih jelas.
2.0.2	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.0	Versi awal.

CloudWatch metrik

Komponen CloudWatch metrik Amazon (`aws.greengrass.Cloudwatch`) menerbitkan metrik khusus dari perangkat inti Greengrass ke Amazon. CloudWatch Komponen ini memungkinkan

komponen untuk mempublikasikan CloudWatch metrik, yang dapat Anda gunakan untuk memantau dan menganalisis lingkungan perangkat inti Greengrass. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch metrik Amazon di Panduan CloudWatch Pengguna Amazon.

Untuk memublikasikan CloudWatch metrik dengan komponen ini, publikasikan pesan ke topik tempat komponen ini berlangganan. Secara default, komponen ini berlangganan topik publikasi/berlangganan lokal `cloudwatch/metric/put`. Anda dapat menentukan topik lain, termasuk topik AWS IoT Core MQTT, saat Anda menerapkan komponen ini.

Komponen ini mengumpulkan metrik yang berada di namespace yang sama dan menerbitkannya secara berkala. CloudWatch

> **Note**
>
> Komponen ini menyediakan fungsionalitas yang mirip dengan konektor CloudWatch metrik di AWS IoT Greengrass V1. Untuk informasi selengkapnya, lihat konektor CloudWatch metrik di Panduan Pengembang AWS IoT Greengrass V1.

Topik

Versi

Komponen ini memiliki versi berikut:

- 3.1.x
- 3.0.x
- 2.1.x
- 2.0.x

Untuk informasi tentang perubahan di setiap versi komponen, lihat changelog.

Tipe

v3.x

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

v2.x

Komponen ini adalah komponen Lambda () `aws.greengrass.lambda`. Inti Greengrass menjalankan fungsi Lambda komponen ini menggunakan komponen peluncur Lambda.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

v3.x

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

v2.x

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

3.x

- Python versi 3.7 diinstal pada perangkat inti dan ditambahkan ke variabel lingkungan PATH.
- Peran perangkat Greengrass harus mengizinkan tindakan `cloudwatch:PutMetricData`, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

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

Untuk informasi selengkapnya, lihat referensi CloudWatch izin Amazon di Panduan CloudWatch Pengguna Amazon.

2.x

- Perangkat inti Anda harus memenuhi persyaratan untuk menjalankan fungsi Lambda. Jika Anda ingin perangkat inti untuk menjalankan fungsi Lambda kontainer, perangkat harus memenuhi persyaratan untuk melakukannya. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.
- Python versi 3.7 diinstal pada perangkat inti dan ditambahkan ke variabel lingkungan PATH.
- Peran perangkat Greengrass harus mengizinkan tindakan `cloudwatch:PutMetricData`, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricData"
      ],
```

```
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Untuk informasi selengkapnya, lihat referensi CloudWatch izin Amazon di Panduan CloudWatch Pengguna Amazon.

- Untuk menerima data keluaran dari komponen ini, Anda harus menggabungkan pemutakhiran konfigurasi berikut untuk komponen router langganan lama (`aws.greengrass.LegacySubscriptionRouter`) saat menerapkan komponen ini. Konfigurasi ini menentukan topik di mana komponen ini menerbitkan tanggapan.

  Legacy subscription router v2.1.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-cloudwatch": {
        "id": "aws-greengrass-cloudwatch",
        "source": "component:aws.greengrass.Cloudwatch",
        "subject": "cloudwatch/metric/put/status",
        "target": "cloud"
      }
    }
  }
  ```

  Legacy subscription router v2.0.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-cloudwatch": {
        "id": "aws-greengrass-cloudwatch",
        "source": "arn:aws:lambda:region:aws:function:aws-greengrass-
  cloudwatch:version",
        "subject": "cloudwatch/metric/put/status",
        "target": "cloud"
      }
    }
  }
  ```

  - Ganti *wilayah* dengan Wilayah AWS yang Anda gunakan.

- Ganti *versi* dengan versi fungsi Lambda yang komponen ini jalankan. Untuk menemukan versi fungsi Lambda, Anda harus melihat resep untuk versi komponen ini yang ingin Anda deploy. Buka halaman detail komponen ini di konsol AWS IoT Greengrass tersebut, dan cari pasangan nilai kunci fungsi Lambda. Pasangan kunci-nilai ini berisi nama dan versi fungsi Lambda.

> **Important**
>
> Anda harus memperbarui versi fungsi Lambda pada router langganan warisan setiap kali Anda men-deploy komponen ini. Hal ini memastikan bahwa Anda menggunakan versi fungsi Lambda yang benar untuk versi komponen yang Anda deploy.

Untuk informasi selengkapnya, lihat Buat deployment.

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Diperlukan	Deskripsi
`monitoring. region .amazonaws.com`	443	Ya	Unggah CloudWatch metrik.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

3.0.0 - 3.1.0

Tabel berikut mencantumkan dependensi untuk versi 3.0.0 hingga 3.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <3.0.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.1.2 and 2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 dan 2.1.3 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.8 - 2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.8 hingga 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.7

Tabel berikut mencantumkan dependensi untuk versi 2.0.7 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.6

Tabel berikut mencantumkan dependensi untuk versi 2.0.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	^2.0.0	Keras

2.0.5

Tabel berikut mencantumkan dependensi untuk versi 2.0.5 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.4

Tabel berikut mencantumkan dependensi untuk versi 2.0.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.3

Tabel berikut mencantumkan dependensi untuk versi 2.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Keras
Peluncur Lambda	>=1.0.0	Keras
Runtime Lambda	>=1.0.0	Lunak
Layanan pertukaran Token	>=1.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

v3.x

`PublishInterval`

(Opsional) Jumlah detik maksimum untuk menunggu sebelum komponen menerbitkan metrik berkelompok untuk namespace tertentu. Untuk mengonfigurasi komponen untuk mempublikasikan metrik saat menerimanya, yang berarti tanpa batching, tentukan `0`.

Komponen dipublikasikan CloudWatch setelah menerima 20 metrik di namespace yang sama atau setelah interval yang Anda tentukan.

> **Note**
>
> Komponen tidak menentukan urutan penerbitan peristiwa.

Nilai ini bisa maksimal 900 detik.

Default: 10 detik

`MaxMetricsToRetain`

(Opsional) Jumlah maksimum metrik di semua namespace untuk disimpan di memori sebelum komponen menggantikannya dengan metrik yang lebih baru.

Batas ini berlaku bila perangkat inti tidak memiliki koneksi ke internet, sehingga komponen mem-buffer metrik untuk dipublikasikan nanti. Ketika buffer penuh, komponen menggantikan metrik tertua dengan yang lebih baru. Metrik dalam namespace tertentu hanya menggantikan metrik dalam namespace yang sama.

> **Note**
>
> Jika proses host untuk komponen terganggu, komponen tidak akan menyimpan metrik. Hal ini dapat terjadi selama deployment atau ketika perangkat inti restart, misalnya.

Nilai ini harus setidaknya 2.000 metrik.

Default: 5.000 metrik

`InputTopic`

(Opsional) Topik yang menjadi langganan komponen untuk menerima pesan. Jika Anda menentukan `true` untuk`PubSubToIoTCore`, Anda dapat menggunakan wildcard MQTT (+ dan #) dalam topik ini.

Default: `cloudwatch/metric/put`

`OutputTopic`

(Opsional) Topik di mana komponen menerbitkan tanggapan status.

Default: `cloudwatch/metric/put/status`

`PubSubToIoTCore`

(Opsional) Nilai string yang menentukan apakah akan mempublikasikan dan berlangganan topik AWS IoT Core MQTT. Nilai yang didukung adalah `true` dan `false`.

Default: `false`

`UseInstaller`

(Opsional) Nilai Boolean yang menentukan apakah akan menggunakan skrip penginstal dalam komponen ini untuk menginstal dependensi SDK komponen ini.

Tetapkan nilai ini `false` jika Anda ingin menggunakan skrip kustom untuk menginstal dependensi, atau jika Anda ingin menyertakan dependensi runtime dalam image Linux yang

sudah dibuat sebelumnya. Untuk menggunakan komponen ini, Anda harus menginstal pustaka berikut, termasuk dependensi apa pun, dan membuatnya tersedia untuk pengguna sistem Greengrass default.

- AWS IoT Device SDK v2 untuk Python
- AWS SDK for Python (Boto3)

Default: `true`

`PublishRegion`

(Opsional) Wilayah AWS Untuk mempublikasikan CloudWatch metrik. Nilai ini menimpa Wilayah default untuk perangkat inti. Parameter ini diperlukan hanya untuk metrik lintas Wilayah.

`accessControl`

(Opsional) Objek yang berisi kebijakan otorisasi yang memungkinkan komponen untuk mempublikasikan dan berlangganan topik yang ditentukan. Jika Anda menentukan nilai kustom untuk `InputTopic` dan`OutputTopic`, Anda harus memperbarui nilai sumber daya dalam objek ini.

Default:

```
{
  "aws.greengrass.ipc.pubsub": {
    "aws.greengrass.Cloudwatch:pubsub:1": {
      "policyDescription": "Allows access to subscribe to input topics.",
      "operations": [
        "aws.greengrass#SubscribeToTopic"
      ],
      "resources": [
        "cloudwatch/metric/put"
      ]
    },
    "aws.greengrass.Cloudwatch:pubsub:2": {
      "policyDescription": "Allows access to publish to output topics.",
      "operations": [
        "aws.greengrass#PublishToTopic"
      ],
      "resources": [
        "cloudwatch/metric/put/status"
      ]
```

```
    }
  },
  "aws.greengrass.ipc.mqttproxy": {
    "aws.greengrass.Cloudwatch:mqttproxy:1": {
      "policyDescription": "Allows access to subscribe to input topics.",
      "operations": [
        "aws.greengrass#SubscribeToIoTCore"
      ],
      "resources": [
        "cloudwatch/metric/put"
      ]
    },
    "aws.greengrass.Cloudwatch:mqttproxy:2": {
      "policyDescription": "Allows access to publish to output topics.",
      "operations": [
        "aws.greengrass#PublishToIoTCore"
      ],
      "resources": [
        "cloudwatch/metric/put/status"
      ]
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi

```
{
  "PublishInterval": 0,
  "PubSubToIoTCore": true
}
```

v2.x

> **Note**
>
> Konfigurasi default komponen ini meliputi parameter fungsi Lambda. Kami sarankan Anda mengedit hanya parameter berikut untuk mengonfigurasi komponen ini pada perangkat Anda.

`lambdaParams`

Sebuah objek yang berisi parameter untuk fungsi Lambda komponen ini. Objek ini berisi informasi berikut:

`EnvironmentVariables`

Sebuah objek yang berisi parameter fungsi Lambda ini. Objek ini berisi informasi berikut:

`PUBLISH_INTERVAL`

(Opsional) Jumlah detik maksimum untuk menunggu sebelum komponen menerbitkan metrik berkelompok untuk namespace tertentu. Untuk mengonfigurasi komponen untuk mempublikasikan metrik saat menerimanya, yang berarti tanpa batching, tentukan `0`.

Komponen dipublikasikan CloudWatch setelah menerima 20 metrik di namespace yang sama atau setelah interval yang Anda tentukan.

> **Note**
>
> Komponen tidak menjamin urutan di mana peristiwa dipublikasikan.

Nilai ini bisa paling banyak 900 detik.

Default: 10 detik

`MAX_METRICS_TO_RETAIN`

(Opsional) Jumlah maksimum metrik di semua namespace untuk disimpan di memori sebelum komponen menggantikannya dengan metrik yang lebih baru.

Batas ini berlaku bila perangkat inti tidak memiliki koneksi ke internet, sehingga komponen mem-buffer metrik untuk dipublikasikan nanti. Ketika buffer penuh, komponen menggantikan metrik tertua dengan yang lebih baru. Metrik dalam namespace tertentu hanya menggantikan metrik dalam namespace yang sama.

> **Note**
>
> Jika proses host untuk komponen terganggu, komponen tidak akan menyimpan metrik. Hal ini dapat terjadi selama deployment atau ketika perangkat inti restart, misalnya.

Nilai ini harus setidaknya 2.000 metrik.

Default: 5.000 metrik

`PUBLISH_REGION`

(Opsional) Wilayah AWS Untuk mempublikasikan CloudWatch metrik. Nilai ini menimpa Wilayah default untuk perangkat inti. Parameter ini diperlukan hanya untuk metrik lintas Wilayah.

`containerMode`

(Opsional) Mode kontainerisasi untuk komponen ini. Pilih dari salah satu pilihan berikut:

- `NoContainer` – Komponen tersebut tidak berjalan di lingkungan waktu aktif terisolasi.
- `GreengrassContainer`— Komponen berjalan di lingkungan runtime yang terisolasi di dalam AWS IoT Greengrass wadah.

Default: `GreengrassContainer`

`containerParams`

(Opsional) Sebuah objek yang berisi parameter kontainer untuk komponen ini. Komponen menggunakan parameter ini jika Anda menentukan `GreengrassContainer` untuk `containerMode`.

Objek ini berisi informasi berikut:

`memorySize`

(Opsional) Jumlah memori (dalam kilobyte) yang akan dialokasikan ke komponen.

Default ke 64 MB (65.535 KB).

`pubsubTopics`

(Opsional) Sebuah objek yang berisi topik di mana komponen berlangganan untuk menerima pesan. Anda dapat menentukan setiap topik dan apakah komponen berlangganan topik MQTT dari AWS IoT Core atau topik penerbitan/langganan lokal.

Objek ini berisi informasi berikut:

`0` - Ini adalah indeks himpunan sebagai string.

Objek yang berisi informasi berikut:

`type`

(Opsional) Jenis olahpesan publikasikan/berlangganan yang digunakan oleh komponen ini untuk berlangganan pesan. Pilih dari salah satu pilihan berikut:

- `PUB_SUB` — Berlangganan pesan publish/subscribe lokal. Jika Anda memilih opsi ini, topik tidak dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Pesan lokal publikasi/berlangganan.
- `IOT_CORE`— Berlangganan pesan AWS IoT Core MQTT. Jika Anda memilih opsi ini, topik dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Terbitkan/berlangganan pesan MQTT AWS IoT Core.

Default: `PUB_SUB`

`topic`

(Opsional) Topik yang menjadi langganan komponen untuk menerima pesan. Jika Anda menentukan `IotCore` untuk `type`, Anda dapat menggunakan wildcard MQTT (+ dan #) dalam topik ini.

Example Contoh: Pembaruan gabungan konfigurasi (mode kontainer)

```
{
  "containerMode": "GreengrassContainer"
}
```

Example Contoh: Pembaruan gabungan konfigurasi (tidak ada mode kontainer)

```
{
  "containerMode": "NoContainer"
}
```

Data input

Komponen ini menerima metrik pada topik berikut dan menerbitkan metrik ke. CloudWatch Secara default, komponen ini berlangganan topik publikasi/berlangganan lokal. Untuk informasi lebih lanjut tentang cara mempublikasikan pesan ke komponen ini dari komponen kustom Anda, lihat Pesan lokal publikasi/berlangganan.

Dimulai dengan versi komponen v3.0.0, Anda dapat mengonfigurasi komponen ini secara opsional untuk berlangganan topik MQTT dengan menyetel parameter konfigurasi ke. `PubSubToIoTCore` `true` Untuk informasi selengkapnya tentang memublikasikan pesan ke topik MQTT di komponen kustom Anda, lihat. Terbitkan/berlangganan pesan MQTT AWS IoT Core

Topik default: `cloudwatch/metric/put`

Pesan menerima properti berikut. Pesan input harus dalam format JSON.

`request`

Metrik dalam pesan ini.

Objek permintaan berisi data metrik untuk dipublikasikan CloudWatch. Nilai metrik harus memenuhi spesifikasi `PutMetricData`operasi.

Jenis: `object` yang berisi informasi 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 sudah dipesan `AWS/`.

Tipe: `string`

Pola yang valid: `[^:].*`

`metricData`

Data untuk metrik tersebut.

Jenis: `object` yang berisi informasi berikut:

`metricName`

Nama metrik.

Tipe: `string`

`value`

Nilai untuk metrik.

> **Note**
>
> CloudWatch menolak nilai yang terlalu kecil atau terlalu besar. Nilai harus antara `8.515920e-109` dan `1.174271e+108` (Basis 10) atau `2e-360` dan `2e360` (Basis 2). CloudWatch tidak mendukung nilai-nilai khusus sepertiNaN,`+Infinity`, dan-`Infinity`.

Tipe: `double`

`dimensions`

(Opsional) Dimensi untuk metrik. Dimensi memberikan informasi tambahan tentang metrik dan datanya. Metrik dapat menentukan hingga 10 dimensi.

Komponen ini secara otomatis menyertakan dimensi bernama`coreName`, di mana nilainya adalah nama perangkat inti.

Jenis: `array` dari objek yang masing-masing berisi informasi berikut:

`name`

(Opsional) Nama dimensi.

Tipe: `string`

`value`

(Opsional) Nilai dimensi.

Tipe: `string`

`timestamp`

(Opsional) Waktu penerimaan data metrik, dinyatakan dalam detik jangka waktu Unix.

Default untuk waktu di mana komponen menerima pesan.

Tipe: `double`

> **Note**
>
> Jika Anda menggunakan antara versi 2.0.3 dan 2.0.7 dari komponen ini, sebaiknya Anda mengambil stempel waktu secara terpisah untuk setiap metrik saat Anda mengirim beberapa metrik dari satu sumber. Jangan menggunakan variabel untuk menyimpan cap waktu.

`unit`

(Opsional) Unit metrik.

Tipe: `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`

Default ke `None`.

> **Note**
>
> Semua kuota yang berlaku untuk CloudWatch `PutMetricData` API berlaku untuk metrik yang Anda terbitkan dengan komponen ini. Kuota 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 kuota CloudWatch layanan di Panduan CloudWatch Pengguna.

Example Contoh input

```
{
  "request": {
    "namespace": "Greengrass",
    "metricData": {
      "metricName": "latency",
      "dimensions": [
        {
          "name": "hostname",
          "value": "test_hostname"
        }
      ],
      "timestamp": 1539027324,
      "value": 123.0,
      "unit": "Seconds"
    }
  }
}
```

Data output

Komponen ini menerbitkan tanggapan sebagai data keluaran pada topik penerbitan/langganan lokal berikut secara default. Untuk informasi lebih lanjut tentang cara mempublikasikan pesan ke komponen ini dari komponen kustom Anda, lihat Pesan lokal publikasi/berlangganan.

Anda dapat mengonfigurasi komponen ini secara opsional untuk memublikasikan ke topik MQTT dengan menyetel parameter konfigurasi kePubSubToIoTCore. `true` Untuk informasi selengkapnya tentang berlangganan pesan tentang topik MQTT di komponen kustom Anda, lihat. Terbitkan/berlangganan pesan MQTT AWS IoT Core

> **Note**
>
> Versi komponen 2.0.x mempublikasikan tanggapan sebagai data keluaran pada topik MQTT secara default. Anda harus menentukan topik sebagai `subject` dalam konfigurasi untuk komponen router langganan lama.

Topik default: `cloudwatch/metric/put/status`

Example Contoh output: Berhasil

Responsnya mencakup namespace data metrik dan `RequestId` bidang dari respons. CloudWatch

```
{
  "response": {
    "cloudwatch_rid": "70573243-d723-11e8-b095-75ff2EXAMPLE",
    "namespace": "Greengrass",
    "status": "success"
  }
}
```

Example Contoh output: Gagal

```
{
  "response" : {
    "namespace": "Greengrass",
    "error": "InvalidInputException",
    "error_message": "cw metric is invalid",
    "status": "fail"
  }
}
```

> **Note**
>
> Jika komponen mendeteksi kesalahan yang dapat dicoba ulang, seperti kesalahan koneksi, ia akan mencoba ulang publikasi di batch berikutnya.

Lisensi

Komponen ini 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/MIT License

Komponen ini dirilis menurut Perjanjian Lisensi Perangkat Lunak Greengrass Core.

File log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.greengrass.Cloudwatch.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.Cloudwatch.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.Cloudwatch.log -Tail 10 -Wait
  ```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

v3.x

Versi	Perubahan
3.1.0	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk konfigurasi proxy jaringan HTTPS. Lihat informasi yang lebih lengkap di Hubungkan pada port 443 atau melalui proksi jaringan dan Aktifkan perangkat inti untuk mempercayai proxy HTTPS.
3.0.0	Versi komponen CloudWatch metrik ini mengharapkan parameter konfigurasi yang berbeda dari versi 2.x. Jika Anda menggunakan konfigurasi non-default untuk versi 2.x, dan Anda ingin meningkatkan dari v2.x ke v3.x, Anda harus memperbarui konfigurasi komponen. Untuk informasi selengkapnya, lihat konfigurasi komponen CloudWatch metrik. Fitur baru • Menambahkan dukungan untuk perangkat inti yang menjalankan Windows. • Mengubah jenis komponen dari komponen Lambda menjadi komponen generik. Komponen ini sekarang tidak lagi bergantung pada komponen router langganan lama untuk membuat langganan. • Menambahkan parameter `InputTopic` konfigurasi baru untuk menentukan topik yang komponen berlangganan untuk menerima pesan. • Menambahkan parameter `OutputTopic` konfigurasi baru untuk menentukan topik yang komponen menerbitkan tanggapan status. • Menambahkan parameter `PubSubToIoTCore` konfigurasi baru untuk menentukan apakah akan mempublikasikan dan berlangganan topik AWS IoT Core MQTT. • Menambahkan parameter `UseInstaller` konfigurasi baru yang memungkinkan Anda menonaktifkan skrip instalasi yang menginstal dependensi komponen secara opsional.

Versi	Perubahan
	Perbaikan bug dan peningkatan Menambahkan dukungan untuk stempel waktu duplikat dalam data input.

v2.x

Versi	Perubahan
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.2	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.1	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.0	Fitur baru • Menambahkan dukungan untuk konfigurasi proxy jaringan HTTPS. Lihat informasi yang lebih lengkap di Hubungkan pada port 443 atau melalui proksi jaringan dan Aktifkan perangkat inti untuk mempercayai proxy HTTPS.
2.0.8	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk stempel waktu duplikat dalam data input. • Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.7	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.6	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.5	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.0.4	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.
2.0.3	Versi awal.

Lihat juga

- Menggunakan CloudWatch metrik Amazon di CloudWatch Panduan Pengguna Amazon
- PutMetricDatadi Referensi CloudWatch API Amazon

AWS IoT Device Defender

AWS IoT Device Defender Component (`aws.greengrass.DeviceDefender`) memberi tahu administrator tentang perubahan status perangkat inti Greengrass. Hal ini dapat membantu mengidentifikasi perilaku yang tidak biasa yang mungkin menunjukkan perangkat yang disusupi. Untuk informasi lebih lanjut, lihat AWS IoT Device Defender dalam Panduan PengembangAWS IoT Core .

Komponen ini membaca metrik sistem pada perangkat inti. Kemudian, ia menerbitkan metrik ke AWS IoT Device Defender. Untuk informasi lebih lanjut tentang cara membaca dan menafsirkan metrik yang dilaporkan oleh komponen ini, lihat Spesifikasi dokumen metrik perangkat dalam Panduan Developer AWS IoT Core .

> **Note**
>
> Komponen ini menyediakan fungsionalitas yang mirip dengan konektor Device Defender di AWS IoT Greengrass V1. Untuk informasi selengkapnya, lihat Konektor Pertahanan Perangkat dalam Panduan Developer AWS IoT Greengrass V1 .

Topik

- Versi
- Tipe
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- Data input
- Data output
- Berkas log lokal

- Lisensi
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 3.1.x
- 3.0.x
- 2.0.x

Untuk informasi tentang perubahan di setiap versi komponen, lihat changelog.

Tipe

v3.x

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

v2.x

Komponen ini adalah komponen Lambda () `aws.greengrass.lambda`. Inti Greengrass menjalankan fungsi Lambda komponen ini menggunakan komponen peluncur Lambda.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

v3.x

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

v2.x

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

v3.x

- Python versi 3.7 diinstal pada perangkat inti dan ditambahkan ke variabel lingkungan PATH.
- AWS IoT Device Defender dikonfigurasi untuk menggunakan fitur Deteksi untuk memantau pelanggaran. Untuk informasi selengkapnya, lihat Deteksi di AWS IoT Core Panduan Developer.

v2.x

- Perangkat inti Anda harus memenuhi persyaratan untuk menjalankan fungsi Lambda. Jika Anda ingin perangkat inti untuk menjalankan fungsi Lambda kontainer, perangkat harus memenuhi persyaratan untuk melakukannya. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.
- Python versi 3.7 diinstal pada perangkat inti dan ditambahkan ke variabel lingkungan PATH.
- AWS IoT Device Defender dikonfigurasi untuk menggunakan fitur Deteksi untuk memantau pelanggaran. Untuk informasi selengkapnya, lihat Deteksi di AWS IoT Core Panduan Developer.
- Pustaka psutil diinstal pada perangkat inti. Versi 5.7.0 adalah versi terkini yang disahkan untuk bekerja dengan komponen.
- Pustaka cbor diinstal pada perangkat inti. Versi 1.0.0 adalah versi terkini yang disahkan untuk bekerja dengan komponen.
- Untuk menerima data keluaran dari komponen ini, Anda harus menggabungkan pemutakhiran konfigurasi berikut untuk komponen router langganan lama (`aws.greengrass.LegacySubscriptionRouter`) saat menerapkan komponen ini. Konfigurasi ini menentukan topik di mana komponen ini menerbitkan tanggapan.

  Legacy subscription router v2.1.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-device-defender": {
        "id": "aws-greengrass-device-defender",
        "source": "component:aws.greengrass.DeviceDefender",
        "subject": "$aws/things/+/defender/metrics/json",
  ```

```
      "target": "cloud"
    }
  }
}
```

Legacy subscription router v2.0.x

```
{
  "subscriptions": {
    "aws-greengrass-device-defender": {
      "id": "aws-greengrass-device-defender",
      "source": "arn:aws:lambda:region:aws:function:aws-greengrass-device-
defender:version",
      "subject": "$aws/things/+/defender/metrics/json",
      "target": "cloud"
    }
  }
}
```

- Ganti *wilayah* dengan Wilayah AWS yang Anda gunakan.
- Ganti *versi* dengan versi fungsi Lambda yang komponen ini jalankan. Untuk menemukan versi fungsi Lambda, Anda harus melihat resep untuk versi komponen ini yang ingin Anda deploy. Buka halaman detail komponen ini di konsol AWS IoT Greengrass tersebut, dan cari pasangan nilai kunci fungsi Lambda. Pasangan kunci-nilai ini berisi nama dan versi fungsi Lambda.

> **Important**
>
> Anda harus memperbarui versi fungsi Lambda pada router langganan warisan setiap kali Anda men-deploy komponen ini. Hal ini memastikan bahwa Anda menggunakan versi fungsi Lambda yang benar untuk versi komponen yang Anda deploy.

Untuk informasi selengkapnya, lihat Buat deployment.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

3.1.1

Tabel berikut mencantumkan dependensi untuk versi 3.1.1 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <3.0.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

3.0.0 - 3.0.2

Tabel berikut mencantumkan dependensi untuk versi 3.0.0 hingga 3.0.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <3.0.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.0.10 and 2.0.11

Tabel berikut mencantumkan dependensi untuk versi 2.0.10 dan 2.0.11 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Keras
Peluncur Lambda	^2.0.0	Keras

Dependensi	Versi yang kompatibel	Jenis dependensi
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.9

Tabel berikut mencantumkan dependensi untuk versi 2.0.9 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.8

Tabel berikut mencantumkan dependensi untuk versi 2.0.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.7

Tabel berikut mencantumkan dependensi untuk versi 2.0.7 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.6

Tabel berikut mencantumkan dependensi untuk versi 2.0.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.5

Tabel berikut mencantumkan dependensi untuk versi 2.0.5 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.4

Tabel berikut mencantumkan dependensi untuk versi 2.0.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.3

Tabel berikut mencantumkan dependensi untuk versi 2.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Keras
Peluncur Lambda	>=1.0.0	Keras
Runtime Lambda	>=1.0.0	Lunak
Layanan pertukaran Token	>=1.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

v3.x

`PublishRetryCount`

Berapa kali publikasi akan dicoba lagi. Fitur ini tersedia dalam versi 3.1.1.

Minimal adalah 0.

Maksimumnya adalah 72.

Default: 5

`SampleIntervalSeconds`

(Opsional) Jumlah waktu dalam hitungan detik antara setiap siklus tempat komponen mengumpulkan dan melaporkan metrik.

Nilai minimum adalah 300 detik (5 menit).

Default: 300 detik

`UseInstaller`

(Opsional) Nilai Boolean yang menentukan apakah akan menggunakan skrip installer dalam komponen ini untuk menginstal dependensi komponen ini.

Tetapkan nilai ini `false` jika Anda ingin menggunakan skrip khusus untuk menginstal dependensi, atau jika Anda ingin menyertakan dependensi runtime dalam image Linux yang sudah dibuat sebelumnya. Untuk menggunakan komponen ini, Anda harus menginstal pustaka berikut, termasuk dependensi apa pun, dan membuatnya tersedia untuk pengguna sistem Greengrass default.

- AWS IoT Device SDK v2 untuk Python
- perpustakaan cbor. Versi 1.0.0 adalah versi terkini yang disahkan untuk bekerja dengan komponen.
- perpustakaan psutil. Versi 5.7.0 adalah versi terkini yang disahkan untuk bekerja dengan komponen.

> **Note**
>
> Jika Anda menggunakan versi 3.0.0 atau 3.0.1 komponen ini pada perangkat inti yang Anda konfigurasikan untuk menggunakan proxy HTTPS, Anda harus menetapkan nilai ini ke. `false` Skrip penginstal tidak mendukung operasi di belakang proxy HTTPS dalam versi komponen ini.

Default: `true`

v2.x

> **Note**
>
> Konfigurasi default komponen ini meliputi parameter fungsi Lambda. Kami sarankan Anda mengedit hanya parameter berikut untuk mengonfigurasi komponen ini pada perangkat Anda.

`lambdaParams`

Sebuah objek yang berisi parameter untuk fungsi Lambda komponen ini. Objek ini berisi informasi berikut:

`EnvironmentVariables`

Sebuah objek yang berisi parameter fungsi Lambda ini. Objek ini berisi informasi berikut:

`PROCFS_PATH`

(Opsional) Jalur ke folder `/proc`.

- Untuk menjalankan komponen ini di kontainer, gunakan nilai default, `/host-proc`. Komponen berjalan dalam kontainer secara default.
- Untuk menjalankan komponen ini ketika tidak ada mode kontainer, tentukan `/proc` untuk parameter ini.

Bawaan: `/host-proc`. Ini adalah jalur default di mana komponen ini memasang folder `/proc` dalam kontainer.

> **Note**
>
> Komponen ini memiliki akses hanya-baca ke folder ini.

`SAMPLE_INTERVAL_SECONDS`

(Opsional) Jumlah waktu dalam hitungan detik antara setiap siklus tempat komponen mengumpulkan dan melaporkan metrik.

Nilai minimum adalah 300 detik (5 menit).

Default: 300 detik

`containerMode`

(Opsional) Mode kontainerisasi untuk komponen ini. Pilih dari salah satu pilihan berikut:

- `GreengrassContainer`— Komponen berjalan di lingkungan runtime yang terisolasi di dalam AWS IoT Greengrass wadah.
- `NoContainer` – Komponen tersebut tidak berjalan di lingkungan waktu aktif terisolasi.

  Jika Anda menentukan opsi ini, Anda harus menentukan `/proc` untuk parameter variabel lingkungan `PROCFS_PATH`.

Default: `GreengrassContainer`

`containerParams`

(Opsional) Sebuah objek yang berisi parameter kontainer untuk komponen ini. Komponen menggunakan parameter ini jika Anda menentukan `GreengrassContainer` untuk `containerMode`.

Objek ini berisi informasi berikut:

`memorySize`

(Opsional) Jumlah memori (dalam kilobyte) yang akan dialokasikan ke komponen.

Defaultnya 50.000 KB.

`pubsubTopics`

(Opsional) Sebuah objek yang berisi topik di mana komponen berlangganan untuk menerima pesan. Anda dapat menentukan setiap topik dan apakah komponen berlangganan topik MQTT dari AWS IoT Core atau topik penerbitan/langganan lokal.

Objek ini berisi informasi berikut:

`0` - Ini adalah indeks himpunan sebagai string.

Objek yang berisi informasi berikut:

`type`

(Opsional) Jenis olahpesan publikasikan/berlangganan yang digunakan oleh komponen ini untuk berlangganan pesan. Pilih dari salah satu pilihan berikut:

- `PUB_SUB` — Berlangganan pesan publish/subscribe lokal. Jika Anda memilih opsi ini, topik tidak dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara

mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Pesan lokal publikasi/berlangganan.

- `IOT_CORE`— Berlangganan pesan AWS IoT Core MQTT. Jika Anda memilih opsi ini, topik dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Terbitkan/berlangganan pesan MQTT AWS IoT Core.

Default: `PUB_SUB`

`topic`

(Opsional) Topik yang menjadi langganan komponen untuk menerima pesan. Jika Anda menentukan `IotCore` untuk `type`, Anda dapat menggunakan wildcard MQTT (+ dan #) dalam topik ini.

Example Contoh: Pembaruan gabungan konfigurasi (mode kontainer)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "PROCFS_PATH": "/host_proc"
    }
  },
  "containerMode": "GreengrassContainer"
}
```

Example Contoh: Pembaruan gabungan konfigurasi (tidak ada mode kontainer)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "PROCFS_PATH": "/proc"
    }
  },
  "containerMode": "NoContainer"
}
```

Data input

Komponen ini tidak menerima pesan sebagai data input.

Data output

Komponen ini menerbitkan metrik keamanan ke topik cadangan berikut untuk. AWS IoT Device Defender Komponen ini diganti *coreDeviceName*dengan nama perangkat inti saat menerbitkan metrik.

Topik (AWS IoT Core MQTT): `$aws/things/coreDeviceName/defender/metrics/json`

Example Contoh Output


```
{
  "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
      }
    }
  }
}
```

Untuk informasi lebih lanjut tentang metrik yang dilaporkan oleh komponen ini, lihat spesifikasi dokumen metrik perangkat dalam Panduan Developer AWS IoT Core .

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.greengrass.DeviceDefender.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.DeviceDefender.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.DeviceDefender.log -Tail 10 -
  Wait
  ```

Lisensi

Komponen ini dirilis menurut Perjanjian Lisensi Perangkat Lunak Greengrass Core.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

v3.x

Versi	Perubahan
3.1.1	Perbaikan bug dan peningkatan • Menambahkan percobaan ulang untuk koneksi klien ketika koneksi gagal pulih setelah pemadaman jaringan. • Menambahkan percobaan ulang yang dapat dikonfigurasi untuk metrik penerbitan.

Versi	Perubahan
3.1.0	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk konfigurasi proxy jaringan HTTPS. Lihat informasi yang lebih lengkap di Hubungkan pada port 443 atau melalui proksi jaringan dan Aktifkan perangkat inti untuk mempercayai proxy HTTPS.
3.0.1	Memperbaiki masalah dengan cara komponen menghitung nilai delta untuk metrik.
3.0.0	**Warning** Versi ini tidak lagi tersedia. Perbaikan dalam versi ini tersedia di versi yang lebih baru dari komponen ini. Versi awal.

v2.x

Versi	Perubahan
2.0.11	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.0.10	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.0.9	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.0.8	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.7	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.6	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.5	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.0.4	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.

Versi	Perubahan
2.0.3	Versi awal.

Spooler disk

Komponen spooler disk (`aws.greengrass.DiskSpooler`) menawarkan opsi penyimpanan persisten untuk pesan yang di-spooled dari perangkat inti Greengrass ke. AWS IoT Core Komponen ini akan menyimpan pesan keluar ini pada disk.

Topik

Versi

Komponen ini memiliki versi berikut:

- 1.0.x

Tipe

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- `storageType`harus diatur `Disk` untuk menggunakan komponen ini. Anda dapat mengatur ini dalam konfigurasi inti Greengrass.
- `maxSizeInBytes`tidak boleh dikonfigurasi agar lebih besar dari ruang yang tersedia di perangkat. Anda dapat mengatur ini dalam konfigurasi inti Greengrass.
- Komponen spooler disk didukung untuk berjalan di VPC.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

1.0.1 – 1.0.3

Tabel berikut mencantumkan dependensi untuk versi 1.0.1 hingga 1.0.3 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.11.0 <2.13.0	Keras

1.0.0

Tabel berikut mencantumkan dependensi untuk versi 1.0.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.11.0 <2.12.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Penggunaan

Untuk menggunakan komponen spooler disk, `aws.greengrass.DiskSpooler` harus dikerahkan.

Untuk mengkonfigurasi dan menggunakan komponen ini, Anda harus mengatur `pluginName` ke`aws.greengrass.DiskSpooler`.

Berkas log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
1.0.3	Perbaikan bug dan peningkatan Meningkatkan kinerja dengan menggunakan kembali koneksi database.
1.0.2	Perbaikan bug dan peningkatan Memperbaiki masalah di mana bidang format pesan MQTT tidak bertahan dalam kasus tertentu.
1.0.1	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
1.0.0	Versi awal.

Manajer aplikasi Docker

Komponen manajer aplikasi Docker (`aws.greengrass.DockerApplicationManager`) memungkinkan AWS IoT Greengrass untuk mengunduh gambar Docker dari pendaftar gambar publik dan pendaftar pribadi yang dihosting di Amazon Elastic Container Registry (Amazon ECR). Ini juga memungkinkan AWS IoT Greengrass untuk mengelola kredensitas secara otomatis untuk mengunduh gambar dengan aman dari repositori pribadi di Amazon ECR.

Ketika Anda mengembangkan komponen kustom yang menjalankan kontainer Docker, sertakan manajer aplikasi Docker sebagai dependensi untuk men-download gambar Docker yang ditetapkan sebagai artefak dalam komponen Anda. Untuk informasi selengkapnya, lihat Jalankan kontainer Docker.

Topik

- Versi
- Tipe
- Sistem operasi
- Persyaratan

- Dependensi
- Konfigurasi
- Berkas log lokal
- Changelog
- Lihat juga

Versi

Komponen ini memiliki versi berikut:

- 2.0.x

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Docker Engine 1.9.1 atau yang lebih baru diinstal pada perangkat inti Greengrass. Versi 20.10 adalah versi terbaru yang diverifikasi untuk bekerja dengan perangkat lunak AWS IoT Greengrass Core. Anda harus menginstal Docker langsung pada perangkat inti sebelum Anda menyebarkan komponen yang menjalankan kontainer Docker.
- Daemon Docker dimulai dan berjalan pada perangkat inti sebelum Anda men-deploy komponen ini.
- Docker gambar disimpan di salah satu sumber gambar yang didukung berikut:

  - Repositori gambar publik dan privat di Amazon Elastic Container Registry (Amazon ECR)
  - Repositori Hub Docker publik
  - Registri Terpercaya Docker publik
- Gambar Docker disertakan sebagai artefak dalam komponen kontainer Docker kustom Anda. Gunakan format URI berikut untuk menentukan gambar Docker Anda:
  - Gambar Amazon ECR privat: `docker:`*`account-id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`repository`*`/`*`image`*`[`*`:tag|@digest`*`]`
  - Citra Amazon ECR publik: `docker:public.ecr.aws/`*`repository`*`/`*`image`*`[`*`:tag|@digest`*`]`
  - Gambar Hub Docker publik: `docker:`*`name`*`[`*`:tag|@digest`*`]`

  Untuk informasi selengkapnya, lihat Jalankan kontainer Docker.

  > **Note**
  >
  > Jika Anda tidak menentukan tanda gambar atau digest gambar di URI artefak untuk gambar, maka manajer aplikasi Docker akan menarik versi terbaru yang tersedia dari gambar tersebut ketika Anda menggunakan komponen kontainer Docker kustom Anda. Untuk memastikan bahwa semua perangkat inti menjalankan versi gambar yang sama, sebaiknya sertakan tag gambar atau diges gambar di URI artefak.
- Pengguna sistem yang menjalankan komponen kontainer Docker harus memiliki izin root atau administrator, atau Anda harus mengonfigurasi Docker untuk menjalankannya sebagai pengguna non-root atau non-admistrator.
  - Pada perangkat Linux, Anda dapat menambahkan pengguna ke `docker` grup untuk memanggil `docker` perintah tanpa`sudo`.
  - Pada perangkat Windows, Anda dapat menambahkan pengguna ke `docker-users` grup untuk memanggil `docker` perintah tanpa hak istimewa adminstrator.

  Linux or Unix

  Untuk menambahkan`ggc_user`, atau pengguna non-root yang Anda gunakan untuk menjalankan komponen kontainer Docker, ke `docker` grup, jalankan perintah berikut.

  ```
  sudo usermod -aG docker ggc_user
  ```

  Untuk informasi selengkapnya, lihat Mengelola Docker sebagai pengguna non-root.

Windows Command Prompt (CMD)

Untuk menambahkan`ggc_user`, atau pengguna yang Anda gunakan untuk menjalankan komponen kontainer Docker, ke `docker-users` grup, jalankan perintah berikut sebagai administrator.

```
net localgroup docker-users ggc_user /add
```

Windows PowerShell

Untuk menambahkan`ggc_user`, atau pengguna yang Anda gunakan untuk menjalankan komponen kontainer Docker, ke `docker-users` grup, jalankan perintah berikut sebagai administrator.

```
Add-LocalGroupMember -Group docker-users -Member ggc_user
```

- Jika Anda mengkonfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan proxy jaringan, Anda harus mengkonfigurasi Docker untuk menggunakan server proxy yang sama.
- Jika gambar Docker Anda disimpan dalam registri privat Amazon ECR, maka Anda harus menyertakan komponen layanan pertukaran token sebagai dependensi dalam komponen kontainer Docker. Juga, peran perangkat Greengrass harus mengizinkan tindakan `ecr:GetAuthorizationToken`, `ecr:BatchGetImage`, dan `ecr:GetDownloadUrlForLayer`, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

- Komponen manajer aplikasi docker didukung untuk berjalan di VPC. Untuk menerapkan komponen ini di VPC, berikut ini diperlukan.
  - Komponen manajer aplikasi docker harus memiliki konektivitas untuk mengunduh gambar. Misalnya, jika Anda menggunakan ECR, Anda harus memiliki konektivitas ke titik akhir berikut.
    - `*.dkr.ecr.region.amazonaws.com`(Titik akhir VPC) `com.amazonaws.region.ecr.dkr`
    - `api.ecr.region.amazonaws.com`(Titik akhir VPC) `com.amazonaws.region.ecr.api`

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Diperlukan	Deskripsi
`ecr.region.amazonaws.com`	443	Tidak	Diperlukan jika Anda mengunduh gambar Docker dari Amazon ECR.

`hub.docker.com`
 	
`registry.hub.docker.com/v1`
	443	Tidak	Diperlukan jika Anda mengunduh gambar Docker dari Docker Hub.

Dependensi

Saat Anda men-deploy komponen, AWS IoT Greengrass juga men-deploy versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua

dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.0.11

Tabel berikut mencantumkan dependensi untuk versi 2.0.11 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.13.0	Lunak

2.0.10

Tabel berikut mencantumkan dependensi untuk versi 2.0.10 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.12.0	Lunak

2.0.9

Tabel berikut mencantumkan dependensi untuk versi 2.0.9 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.11.0	Lunak

2.0.8

Tabel berikut mencantumkan dependensi untuk versi 2.0.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.10.0	Lunak

2.0.7

Tabel berikut mencantumkan dependensi untuk versi 2.0.7 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.9.0	Lunak

2.0.6

Tabel berikut mencantumkan dependensi untuk versi 2.0.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.8.0	Lunak

2.0.5

Tabel berikut mencantumkan dependensi untuk versi 2.0.5 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.7.0	Lunak

2.0.4

Tabel berikut mencantumkan dependensi untuk versi 2.0.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.6.0	Lunak

2.0.3

Tabel berikut mencantumkan dependensi untuk versi 2.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.5.0	Lunak

2.0.2

Tabel berikut mencantumkan dependensi untuk versi 2.0.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.4.0	Lunak

2.0.1

Tabel berikut mencantumkan dependensi untuk versi 2.0.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.3.0	Lunak

2.0.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.2.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

Berkas log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.0.11	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.0.10	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.0.9	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.0.8	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.0.7	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.

Versi	Perubahan
2.0.6	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.0.5	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.0.4	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.0.0	Versi awal.

Lihat juga

- Jalankan kontainer Docker

Konektor tepi untuk Kinesis Video Streams

Konektor tepi untuk `aws.iot.EdgeConnectorForKVS` komponen Kinesis Video Streams () membaca umpan video dari kamera lokal dan menerbitkan aliran ke Kinesis Video Streams. Anda dapat mengonfigurasi komponen ini untuk membaca umpan video dari kamera Internet Protocol (IP) menggunakan Real Time Streaming Protocol (RTSP). Kemudian, Anda dapat mengatur dasbor di Grafana Terkelola Amazon atau server Grafana lokal untuk memantau dan berinteraksi dengan aliran video.

Anda dapat mengintegrasikan komponen ini AWS IoT TwinMaker untuk menampilkan dan mengontrol aliran video di dasbor Grafana. AWS IoT TwinMaker adalah AWS layanan yang memungkinkan Anda membangun kembar digital operasional sistem fisik. Anda dapat menggunakan AWS IoT TwinMaker untuk memvisualisasikan data dari sensor, kamera, dan aplikasi perusahaan bagi Anda untuk melacak pabrik fisik, bangunan, atau pabrik industri Anda. Anda juga dapat menggunakan data ini untuk memantau operasi, mendiagnosis kesalahan, dan memperbaiki kesalahan. Untuk informasi lebih lanjut, lihat Apa itu AWS IoT TwinMaker? dalam AWS IoT TwinMaker User Guide.

Komponen ini menyimpan konfigurasinya di AWS IoT SiteWise, yang merupakan AWS layanan yang memodelkan dan menyimpan data industri. Dalam AWS IoT SiteWise, aset mewakili objek seperti perangkat, peralatan, atau kelompok objek lain. Untuk mengonfigurasi dan menggunakan komponen ini, Anda membuat AWS IoT SiteWise aset untuk setiap perangkat inti Greengrass dan untuk setiap kamera IP yang terhubung ke setiap perangkat inti. Setiap aset memiliki properti yang Anda konfigurasikan untuk mengontrol fitur, seperti streaming langsung, unggahan sesuai permintaan, dan caching lokal. Untuk menentukan URL untuk setiap kamera, Anda membuat rahasia AWS Secrets Manager yang berisi URL kamera. Jika kamera memerlukan otentikasi, Anda juga menentukan nama pengguna dan kata sandi di URL. Kemudian, Anda menentukan rahasia itu di properti aset untuk kamera IP.

Komponen ini mengunggah aliran video setiap kamera ke aliran video Kinesis. Anda menentukan nama aliran video Kinesis tujuan dalam konfigurasi AWS IoT SiteWise aset untuk setiap kamera. Jika aliran video Kinesis tidak ada, komponen ini membuatnya untuk Anda.

AWS IoT TwinMaker menyediakan skrip yang dapat Anda jalankan untuk membuat AWS IoT SiteWise aset dan rahasia Secrets Manager ini. Untuk informasi selengkapnya tentang cara membuat sumber daya ini, serta cara menginstal, mengonfigurasi, dan menggunakan komponen ini, lihat integrasi AWS IoT TwinMaker video di Panduan AWS IoT TwinMaker Pengguna.

> **Note**
>
> Konektor tepi untuk komponen Kinesis Video Streams hanya tersedia dalam hal berikut: Wilayah AWS
>
> - AS Timur (Virginia Utara)
> - AS Barat (Oregon)
> - Eropa (Frankfurt)
> - Eropa (Irlandia)
> - Asia Pasifik (Singapura)

Topik

- Dependensi
- Konfigurasi
- Lisensi
- Penggunaan
- Berkas log lokal
- Changelog
- Lihat juga

Versi

Komponen ini memiliki versi berikut:

- 1.0.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Anda dapat menerapkan komponen ini hanya ke perangkat inti tunggal, karena konfigurasi komponen harus unik untuk setiap perangkat inti. Anda tidak dapat menerapkan komponen ini ke grup perangkat inti.
- GStreamer 1.18.4 atau yang lebih baru diinstal pada perangkat inti. Untuk informasi selengkapnya, lihat Menginstal GStreamer.

  Pada perangkat dengan`apt`, Anda dapat menjalankan perintah berikut untuk menginstal GStreamer.

```
sudo apt install -y libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
 gstreamer1.0-plugins-base-apps
sudo apt install -y gstreamer1.0-libav
sudo apt install -y gstreamer1.0-plugins-bad gstreamer1.0-plugins-good gstreamer1.0-
plugins-ugly gstreamer1.0-tools
```

- AWS IoT SiteWise Aset untuk setiap perangkat inti. AWS IoT SiteWise Aset ini mewakili perangkat inti. Untuk informasi selengkapnya tentang cara membuat aset ini, lihat integrasi AWS IoT TwinMaker video di Panduan AWS IoT TwinMaker Pengguna.
- AWS IoT SiteWise Aset untuk setiap kamera IP yang Anda sambungkan ke setiap perangkat inti. AWS IoT SiteWise Aset ini mewakili kamera yang mengalirkan video ke setiap perangkat inti. Setiap aset kamera harus dikaitkan dengan aset untuk perangkat inti yang terhubung ke kamera. Aset kamera memiliki properti yang dapat Anda konfigurasikan untuk menentukan aliran video Kinesis, rahasia otentikasi, dan parameter streaming video. Untuk informasi selengkapnya tentang cara membuat dan mengonfigurasi aset kamera, lihat integrasi AWS IoT TwinMaker video di Panduan AWS IoT TwinMaker Pengguna.
- AWS Secrets Manager Rahasia untuk setiap kamera IP. Rahasia ini harus mendefinisikan pasangan kunci-nilai, di mana kuncinya`RTSPStreamUrl`, dan nilainya adalah URL untuk kamera. Jika kamera memerlukan otentikasi, sertakan nama pengguna dan kata sandi di URL ini. Anda dapat menggunakan skrip untuk membuat rahasia saat Anda membuat sumber daya yang dibutuhkan komponen ini. Untuk informasi selengkapnya, lihat integrasi AWS IoT TwinMaker video di Panduan AWS IoT TwinMaker Pengguna.

  Anda juga dapat menggunakan konsol Secrets Manager dan API untuk membuat rahasia tambahan. Untuk informasi selengkapnya, lihat Membuat rahasia di Panduan AWS Secrets Manager Pengguna.
- Peran pertukaran token Greengrass harus mengizinkan tindakan AWS Secrets Manager berikut,, dan AWS IoT SiteWise Kinesis Video Streams, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

> **Note**
>
> Kebijakan contoh ini memungkinkan perangkat untuk mendapatkan nilai rahasia bernama **`IPCamera1Url`** dan**`IPCamera2Url`**. Saat Anda mengonfigurasi setiap kamera IP, Anda menentukan rahasia yang berisi URL untuk kamera itu. Jika kamera memerlukan otentikasi, Anda juga menentukan nama pengguna dan kata sandi di URL. Peran

pertukaran token perangkat inti harus memungkinkan akses ke rahasia untuk setiap kamera IP untuk terhubung.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:secretsmanager:region:account-id:secret:IPCamera1Url",
        "arn:aws:secretsmanager:region:account-id:secret:IPCamera2Url"
      ]
    },
    {
      "Action": [
        "iotsitewise:BatchPutAssetPropertyValue",
        "iotsitewise:DescribeAsset",
        "iotsitewise:DescribeAssetModel",
        "iotsitewise:DescribeAssetProperty",
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:ListAssetRelationships",
        "iotsitewise:ListAssets",
        "iotsitewise:ListAssociatedAssets",
        "kinesisvideo:CreateStream",
        "kinesisvideo:DescribeStream",
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:PutMedia",
        "kinesisvideo:TagStream"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    }
  ]
}
```

> **Note**
>
> Jika Anda menggunakan AWS Key Management Service kunci yang dikelola pelanggan untuk mengenkripsi rahasia, peran perangkat juga harus mengizinkan `kms:Decrypt` tindakan.

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Wajib	Deskripsi
`kinesisvideo.` *`region`*`.amazonaws.com`	443	Ya	Unggah data ke Kinesis Video Streams.
`data.iotsitewise.` *`region`*`.amazonaws.com`	443	Ya	Publikasikan metadata streaming video ke. AWS IoT SiteWise
`secretsmanager.` *`region`*`.amazonaws.com`	443	Ya	Unduh rahasia URL kamera ke perangkat inti.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

Tabel berikut mencantumkan dependensi untuk versi 1.0.0 hingga 1.0.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	>=2.0.3	Keras
Manajer aliran	>=2.0.9	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`SiteWiseAssetIdForHub`

ID AWS IoT SiteWise aset yang mewakili perangkat inti ini. Untuk informasi selengkapnya tentang cara membuat aset ini dan menggunakannya untuk berinteraksi dengan komponen ini, lihat integrasi AWS IoT TwinMaker video di Panduan AWS IoT TwinMaker Pengguna.

Example Contoh: Pembaruan gabungan konfigurasi

```
{
  "SiteWiseAssetIdForHub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

Lisensi

Komponen ini mencakup perangkat lunak/lisensi pihak ketiga berikut:

- Penjadwal Pekerjaan Kuarsa/Lisensi Apache 2.0
- Binding Java untuk GStreamer 1.x/GNU Lesser General Public License v3.0

Penggunaan

Untuk mengonfigurasi dan berinteraksi dengan komponen ini, Anda dapat mengatur properti pada AWS IoT SiteWise aset yang mewakili perangkat inti dan kamera IP tempat ia terhubung. Anda juga dapat memvisualisasikan dan berinteraksi dengan aliran video di dasbor Grafana melalui. AWS IoT TwinMaker Untuk informasi selengkapnya, lihat integrasi AWS IoT TwinMaker video di Panduan AWS IoT TwinMaker Pengguna.

Berkas log lokal

Komponen ini menggunakan file log berikut.

```
/greengrass/v2/logs/aws.iot.EdgeConnectorForKVS.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* dengan jalur ke folder AWS IoT Greengrass root.

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.EdgeConnectorForKVS.log
  ```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
1.0.5	Perbaikan bug umum dan perbaikan.
1.0.4	Perbaikan bug dan peningkatan • Memperbaiki masalah yang menyebabkan pengunggahan langsung berhenti.
1.0.3	Perbaikan bug umum dan perbaikan.

Versi	Perubahan
1.0.1	Perbaikan bug umum dan perbaikan.
1.0.0	Versi awal.

Lihat juga

- Apa itu AWS IoT TwinMaker? di Panduan AWS IoT TwinMaker Pengguna
- AWS IoT TwinMaker integrasi video dalam Panduan AWS IoT TwinMaker Pengguna
- Apa itu AWS IoT SiteWise? di Panduan AWS IoT SiteWise Pengguna
- Memperbarui nilai atribut dalam Panduan AWS IoT SiteWise Pengguna
- Apa itu AWS Secrets Manager? dalam AWS Secrets Manager Panduan Penggguna
- Membuat dan mengelola rahasia di Panduan AWS Secrets Manager Pengguna

Greengrass CLI

Komponen Greengrass CLI (`aws.greengrass.Cli`) menyediakan antarmuka baris perintah lokal yang dapat Anda gunakan pada perangkat inti untuk mengembangkan dan debug komponen secara lokal. Greengrass CLI memungkinkan Anda untuk membuat deployment lokal dan me-restart komponen pada perangkat inti, misalnya.

Anda dapat menginstal komponen ini ketika Anda menginstal perangkat lunak AWS IoT Greengrass Core. Untuk informasi selengkapnya, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2.

> **Important**
>
> Kami menyarankan Anda menggunakan komponen ini hanya di lingkungan pengembangan, bukan lingkungan produksi. Komponen ini menyediakan akses ke informasi dan operasi yang biasanya tidak Anda perlukan di lingkungan produksi. Ikuti prinsip hak istimewa paling sedikit dengan menerapkan komponen ini hanya ke perangkat inti di mana Anda membutuhkannya.

Setelah Anda menginstal komponen ini, jalankan perintah berikut untuk melihat dokumentasi bantuan. Ketika komponen ini diinstal, komponen tersebut menambahkan tautan simbolik ke

`greengrass-cli` dalam folder */greengrass/v2*`/bin`. Anda dapat menjalankan Greengrass CLI dari jalur ini atau menambahkannya ke variabel lingkungan `PATH` Anda untuk menjalankan `greengrass-cli` tanpa jalur absolutnya.

Linux or Unix

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

Windows

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

Perintah berikut me-restart komponen bernama `com.example.HelloWorld`, misalnya.

Linux or Unix

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

Windows

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

Untuk informasi selengkapnya, lihat Antarmuka Baris Perintah Greengrass.

Topik

- Versi
- Tipe
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- Berkas log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.12.x
- 2.11.x
- 2.10.x
- 2.9.x
- 2.8.x
- 2.7.x
- 2.6.x
- 2.5.x
- 2.4.x
- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Anda harus diberi wewenang untuk menggunakan CLI Greengrass untuk berinteraksi dengan perangkat lunak Core. AWS IoT Greengrass Lakukan salah satu langkah berikut untuk menggunakan Greengrass CLI:
  - Gunakan pengguna sistem yang menjalankan perangkat lunak AWS IoT Greengrass Core.
  - Gunakan pengguna dengan izin root atau adminstratif. Pada perangkat inti Linux, Anda dapat menggunakan `sudo` untuk mendapatkan izin root.
  - Gunakan pengguna sistem yang berada dalam grup yang Anda tentukan dalam parameter `AuthorizedPosixGroups` atau `AuthorizedWindowsGroups` konfigurasi saat Anda menerapkan komponen. Untuk informasi selengkapnya, lihat konfigurasi komponen CLI Greengrass.
- Komponen CLI Greengrass didukung untuk berjalan di VPC.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.12.0 – 2.12.6

Tabel berikut mencantumkan dependensi untuk versi 2.12.0 hingga 2.12.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.12.0 <2.13.0	Lunak

2.11.0 – 2.11.3

Tabel berikut mencantumkan dependensi untuk versi 2.11.0 hingga 2.11.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.11.0 <2.12.0	Lunak

2.10.0 – 2.10.3

Tabel berikut mencantumkan dependensi untuk versi 2.10.0 hingga 2.10.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.11.0	Lunak

2.9.0 – 2.9.6

Tabel berikut mencantumkan dependensi untuk versi 2.9.0 hingga 2.9.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.10.0	Lunak

2.8.0 – 2.8.1

Tabel berikut mencantumkan dependensi untuk versi 2.8.0 dan 2.8.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.9.0	Lunak

2.7.0

Tabel berikut mencantumkan dependensi untuk versi 2.7.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.8.0	Lunak

2.6.0

Tabel berikut mencantumkan dependensi untuk versi 2.6.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.7.0	Lunak

2.5.0 – 2.5.6

Tabel berikut mencantumkan dependensi untuk versi 2.5.0 hingga 2.5.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.6.0	Lunak

2.4.0

Tabel berikut mencantumkan dependensi untuk versi 2.4.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.5.0	Lunak

2.3.0

Tabel berikut mencantumkan dependensi untuk versi 2.3.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.4.0	Lunak

2.2.0

Tabel berikut mencantumkan dependensi untuk versi 2.2.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.3.0	Lunak

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.2.0	Lunak

2.0.x

Tabel berikut mencantumkan dependensi untuk versi 2.0.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.1.0	Lunak

> **Note**
>
> Versi kompatibel minimum dari nukleus Greengrass sesuai dengan versi patch komponen Greengrass CLI.

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

2.5.x - 2.12.x

`AuthorizedPosixGroups`

(Opsional) String yang berisi daftar kelompok sistem yang dipisahkan dengan koma. Anda mengizinkan grup sistem ini untuk menggunakan CLI Greengrass untuk berinteraksi dengan perangkat lunak Inti. AWS IoT Greengrass Anda dapat menentukan nama grup atau ID grup. Misalnya, `group1,1002,group3` mengotorisasi tiga grup sistem (`group1`, `1002`, dan `group3`) untuk menggunakan Greengrass CLI.

Jika Anda tidak menentukan grup apa pun untuk diotorisasi, Anda dapat menggunakan Greengrass CLI sebagai `sudo` pengguna root () atau sebagai pengguna sistem yang menjalankan perangkat lunak Core. AWS IoT Greengrass

`AuthorizedWindowsGroups`

(Opsional) String yang berisi daftar kelompok sistem yang dipisahkan dengan koma. Anda mengizinkan grup sistem ini untuk menggunakan CLI Greengrass untuk berinteraksi dengan perangkat lunak Inti. AWS IoT Greengrass Anda dapat menentukan nama grup atau ID grup. Misalnya, `group1,1002,group3` mengotorisasi tiga grup sistem (`group1`, `1002`, dan `group3`) untuk menggunakan Greengrass CLI.

Jika Anda tidak menentukan grup apa pun untuk diotorisasi, Anda dapat menggunakan CLI Greengrass sebagai administrator atau sebagai pengguna sistem yang menjalankan perangkat lunak Core. AWS IoT Greengrass

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk mengotorisasi tiga grup sistem POSIX (`group1`,`1002`, dan`group3`) dan dua kelompok pengguna Windows (`Device Operators`dan`QA Engineers`) untuk menggunakan CLI Greengrass.

```
{
  "AuthorizedPosixGroups": "group1,1002,group3",
  "AuthorizedWindowsGroups": "Device Operators,QA Engineers"
}
```

2.4.x - 2.0.x

`AuthorizedPosixGroups`

(Opsional) String yang berisi daftar kelompok sistem yang dipisahkan dengan koma. Anda mengizinkan grup sistem ini untuk menggunakan CLI Greengrass untuk berinteraksi dengan perangkat lunak Inti. AWS IoT Greengrass Anda dapat menentukan nama grup atau ID grup. Misalnya, `group1,1002,group3` mengotorisasi tiga grup sistem (`group1`, `1002`, dan `group3`) untuk menggunakan Greengrass CLI.

Jika Anda tidak menentukan grup apa pun untuk diotorisasi, Anda dapat menggunakan Greengrass CLI sebagai `sudo` pengguna root () atau sebagai pengguna sistem yang menjalankan perangkat lunak Core. AWS IoT Greengrass

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut akan menentukan untuk mengotorisasi tiga kelompok sistem (`group1`, `1002`, dan `group3`) untuk menggunakan Greengrass CLI.

```
{
  "AuthorizedPosixGroups": "group1,1002,group3"
}
```

Berkas log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.12.6	Versi diperbarui untuk Greengrass nucleus versi 2.12.6 rilis.
2.12.5	Versi diperbarui untuk Greengrass nucleus versi 2.12.5 rilis.
2.12.4	Versi diperbarui untuk Greengrass nucleus versi 2.12.4 rilis.
2.12.3	**Warning** Versi ini tidak lagi tersedia. Perbaikan dalam versi ini tersedia di versi yang lebih baru dari komponen ini. Versi diperbarui untuk Greengrass nucleus versi 2.12.3 rilis.
2.12.2	Versi diperbarui untuk Greengrass nucleus versi 2.12.2 rilis.
2.12.1	Versi diperbarui untuk Greengrass nucleus versi 2.12.1 rilis.
2.12.0	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.

Versi	Perubahan
2.11.3	Versi diperbarui untuk Greengrass nucleus versi 2.11.3 rilis.
2.11.2	Versi diperbarui untuk Greengrass nucleus versi 2.11.2 rilis.
2.11.1	Versi diperbarui untuk Greengrass nucleus versi 2.11.1 rilis.
2.11.0	Fitur baru • Memungkinkan Anda membatalkan penerapan lokal. • Memungkinkan Anda mengonfigurasi kebijakan penanganan kegagalan untuk penerapan lokal. • Meningkatkan pelaporan status penyebaran terperinci.
2.10.3	Versi diperbarui untuk Greengrass nucleus versi 2.10.3 rilis.
2.10.2	Versi diperbarui untuk Greengrass nucleus versi 2.10.2 rilis.
2.10.1	Versi diperbarui untuk Greengrass nucleus versi 2.10.1 rilis.
2.10.0	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.9.6	Versi diperbarui untuk Greengrass nucleus versi 2.9.6 rilis.
2.9.5	Versi diperbarui untuk Greengrass nucleus versi 2.9.5 rilis.
2.9.4	Versi diperbarui untuk Greengrass nucleus versi 2.9.4 rilis.
2.9.3	Versi diperbarui untuk Greengrass nucleus versi 2.9.3 rilis.
2.9.2	Versi diperbarui untuk Greengrass nucleus versi 2.9.2 rilis.
2.9.1	Versi diperbarui untuk Greengrass nucleus versi 2.9.1 rilis.
2.9.0	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.8.1	Versi diperbarui untuk Greengrass nucleus versi 2.8.1 rilis.
2.8.0	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.7.0	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.

Versi	Perubahan
2.6.0	Fitur baru • Menambahkan dukungan untuk komponen khusus untuk memanggil operasi komunikasi antarproses (IPC) yang digunakan CLI Greengrass. Anda dapat menggunakan operasi IPC ini untuk mengelola penerapan lokal, melihat detail komponen, dan membuat kata sandi yang dapat Anda gunakan untuk masuk ke konsol debug lokal. Untuk informasi selengkapnya, lihat IPC: Mengelola penerapan dan komponen lokal. Perbaikan bug dan peningkatan • Peningkatan dan perbaikan kecil tambahan.
2.5.6	Versi diperbarui untuk Greengrass nucleus versi 2.5.6 rilis.
2.5.5	Versi diperbarui untuk Greengrass nucleus versi 2.5.5 rilis.
2.5.4	Versi diperbarui untuk Greengrass nucleus versi 2.5.4 rilis.
2.5.3	Versi diperbarui untuk Greengrass nucleus versi 2.5.3 rilis.
2.5.2	Versi diperbarui untuk Greengrass nucleus versi 2.5.2 rilis.
2.5.1	Versi diperbarui untuk Greengrass nucleus versi 2.5.1 rilis.
2.5.0	Fitur baru • Menambahkan dukungan untuk perangkat inti yang menjalankan Windows. • Menambahkan parameter `AuthorizedWindowsGroups` konfigurasi baru yang dapat Anda tentukan untuk mengotorisasi grup sistem untuk menggunakan CLI Greengrass pada perangkat Windows. • Menambahkan `windowsUser` parameter untuk penerapan lokal. Anda dapat menggunakan parameter ini menentukan pengguna yang akan digunakan untuk menjalankan komponen pada perangkat inti Windows.

Versi	Perubahan
2.4.0	Fitur baru • Menambahkan dukungan untuk batas sumber daya sistem. Saat Anda membuat penyebaran lokal, Anda dapat mengonfigurasi jumlah maksimum penggunaan CPU dan RAM yang dapat digunakan oleh setiap proses komponen pada perangkat inti. Untuk informasi selengkap nya, lihat Konfigurasikan batas sumber daya sistem untuk komponen dan perintah deployment create.
2.3.0	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.2.0	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.1.0	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.
2.0.5	Versi yang diperbarui untuk rilis inti Greengrass versi 2.0.5.
2.0.4	Versi yang diperbarui untuk rilis inti Greengrass versi 2.0.4.
2.0.3	Versi awal.

Detektor IP

Komponen detektor IP (`aws.greengrass.clientdevices.IPDetector`) melakukan hal berikut:

- Memantau informasi konektivitas jaringan perangkat inti Greengrass. Informasi ini mencakup titik akhir jaringan perangkat inti dan port tempat broker MQTT beroperasi.
- Memperbarui informasi konektivitas perangkat inti di layanan AWS IoT Greengrass cloud.

Perangkat klien dapat menggunakan penemuan cloud Greengrass untuk mengambil informasi konektivitas perangkat inti terkait. Kemudian, perangkat klien dapat mencoba untuk menyambung ke setiap perangkat inti sampai berhasil terhubung.

> **Note**
>
> Perangkat klien adalah perangkat IoT lokal yang terhubung ke perangkat inti Greengrass untuk mengirim pesan MQTT dan data yang akan diproses. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Komponen detektor IP menggantikan informasi konektivitas perangkat inti yang ada dengan informasi dideteksinya. Karena komponen ini menghapus informasi yang ada, Anda dapat menggunakan komponen detektor IP, atau secara manual mengelola informasi konektivitas.

> **Note**
>
> Komponen detektor IP hanya mendeteksi alamat IPv4.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Peran layanan Greengrass harus dikaitkan dengan Akun AWS Anda dan mengizinkan serta izin. `iot:GetThingShadow` `iot:UpdateThingShadow`
- AWS IoT Kebijakan perangkat inti harus mengizinkan `greengrass:UpdateConnectivityInfo` izin. Lihat informasi yang lebih lengkap di Kebijakan AWS IoT untuk operasi bidang data dan Kebijakan AWS IoT minimal untuk mendukung perangkat klien.
- Jika Anda mengonfigurasi komponen broker MQTT perangkat inti untuk menggunakan port selain port default 8883, Anda harus menggunakan detektor IP v2.1.0 atau yang lebih baru. Konfigurasikan untuk melaporkan port tempat broker beroperasi.
- Jika Anda memiliki penataan jaringan yang kompleks, komponen detektor IP mungkin tidak dapat mengidentifikasi titik akhir di mana perangkat klien dapat menyambung ke perangkat inti. Jika komponen detektor IP tidak dapat mengelola titik akhir, Anda harus secara manual mengelola titik akhir perangkat inti sebagai gantinya. Sebagai contoh, jika perangkat inti berada di belakang router yang meneruskan port broker MQTT ke sana, Anda harus menentukan alamat IP router sebagai titik akhir untuk perangkat inti. Untuk informasi selengkapnya, lihat Kelola titik akhir perangkat inti.
- Komponen detektor IP didukung untuk berjalan di VPC.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.8 – 2.1.9

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 dan 2.1.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.13.0	Lunak

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.12.0	Lunak

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.11.0	Lunak

2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.10.0	Lunak

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.9.0	Lunak

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.8.0	Lunak

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.7.0	Lunak

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.6.0	Lunak

2.1.0 and 2.0.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 dan 2.0.2 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.5.0	Lunak

2.0.1

Tabel berikut mencantumkan dependensi untuk versi 2.0.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.4.0	Lunak

2.0.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.3.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

2.1.x

`defaultPort`

(Opsional) Port broker MQTT untuk melaporkan kapan komponen ini mendeteksi alamat IP. Anda harus menentukan parameter ini jika Anda mengonfigurasi broker MQTT untuk menggunakan port yang berbeda dari port default 8883.

Default: `8883`

`includeIPv4LoopbackAddrs`

(Opsional) Anda dapat mengaktifkan opsi ini untuk mendeteksi dan melaporkan alamat loopback IPv4. Ini adalah alamat IP, seperti`localhost`, di mana perangkat dapat berkomunikasi dengan dirinya sendiri. Gunakan opsi ini di lingkungan pengujian tempat perangkat inti dan perangkat klien berjalan pada sistem yang sama.

Default: `false`

`includeIPv4LinkLocalAddrs`

(Opsional) Anda dapat mengaktifkan opsi ini untuk mendeteksi dan melaporkan alamat link-lokal IPv4. Gunakan opsi ini jika jaringan perangkat inti tidak memiliki Dynamic Host Configuration Protocol (DHCP) atau alamat IP yang ditetapkan secara statis.

Default: `false`

2.0.x

`includeIPv4LoopbackAddrs`

(Opsional) Anda dapat mengaktifkan opsi ini untuk mendeteksi dan melaporkan alamat loopback IPv4. Ini adalah alamat IP, seperti`localhost`, di mana perangkat dapat berkomunikasi dengan dirinya sendiri. Gunakan opsi ini di lingkungan pengujian tempat perangkat inti dan perangkat klien berjalan pada sistem yang sama.

Default: `false`

`includeIPv4LinkLocalAddrs`

(Opsional) Anda dapat mengaktifkan opsi ini untuk mendeteksi dan melaporkan alamat link-lokal IPv4. Gunakan opsi ini jika jaringan perangkat inti tidak memiliki Dynamic Host Configuration Protocol (DHCP) atau alamat IP yang ditetapkan secara statis.

Default: `false`

Berkas log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.9	Perbaikan bug dan peningkatan • Menyesuaikan langkah yang diperoleh IP untuk hanya mengirim log pada tingkat log debug.
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.

Versi	Perubahan
2.1.5	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.2	Perbaikan bug dan peningkatan • Meningkatkan pesan kesalahan yang dicatat komponen ini dalam skenario tertentu. • Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.1	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.0	Perbaikan • Menambahkan `defaultPort` parameter, yang memungkinkan Anda untuk menggunakan port broker MQTT non-default. • Pembaruan untuk membuat pesan log lebih jelas.
2.0.2	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.0	Versi awal.

Firehose

Komponen Firehose (`aws.greengrass.KinesisFirehose`) menerbitkan data melalui aliran pengiriman Amazon Data Firehose ke tujuan, seperti Amazon S3, Amazon Redshift, dan Amazon Service. OpenSearch Untuk informasi selengkapnya, lihat Apa itu Amazon Data Firehose? di Panduan Pengembang Firehose Data Amazon.

Untuk mempublikasikan aliran pengiriman Kinesis dengan komponen ini, publikasikan pesan ke topik yang berlangganan komponen ini. Secara default, komponen ini berlangganan topik publish/subscribe `kinesisfirehose/message` dan `kinesisfirehose/message/binary/#` lokal. Anda dapat menentukan topik lain, termasuk topik AWS IoT Core MQTT, saat Anda menerapkan komponen ini.

> **Note**
>
> Komponen ini menyediakan fungsionalitas yang mirip dengan konektor Firehose di AWS IoT Greengrass V1. Untuk informasi selengkapnya, lihat Konektor Firehose di Panduan Pengembang AWS IoT Greengrass V1.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen Lambda () `aws.greengrass.lambda`. Inti Greengrass menjalankan fungsi Lambda komponen ini menggunakan komponen peluncur Lambda.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Anda harus memenuhi persyaratan untuk menjalankan fungsi Lambda. Jika Anda ingin perangkat inti untuk menjalankan fungsi Lambda kontainer, perangkat harus memenuhi persyaratan untuk melakukannya. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.
- Python versi 3.7 diinstal pada perangkat inti dan ditambahkan ke variabel lingkungan PATH.
- Peran perangkat Greengrass harus mengizinkan tindakan `firehose:PutRecord`, dan `firehose:PutRecordBatch`, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:firehose:region:account-id:deliverystream/stream-name"
      ]
    }
  ]
}
```

  Anda dapat secara dinamis menimpa aliran pengiriman default dalam muatan pesan masukan untuk komponen ini. Jika aplikasi Anda menggunakan fitur ini, kebijakan IAM harus mencakup semua aliran target sebagai sumber daya. Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya (misalnya, dengan menggunakan skema penamaan wildcard *).
- Untuk menerima data keluaran dari komponen ini, Anda harus menggabungkan pemutakhiran konfigurasi berikut untuk komponen router langganan lama (`aws.greengrass.LegacySubscriptionRouter`) saat menerapkan komponen ini. Konfigurasi ini menentukan topik di mana komponen ini menerbitkan tanggapan.

Legacy subscription router v2.1.x

```
{
  "subscriptions": {
    "aws-greengrass-kinesisfirehose": {
      "id": "aws-greengrass-kinesisfirehose",
      "source": "component:aws.greengrass.KinesisFirehose",
      "subject": "kinesisfirehose/message/status",
      "target": "cloud"
    }
  }
}
```

Legacy subscription router v2.0.x

```
{
  "subscriptions": {
    "aws-greengrass-kinesisfirehose": {
      "id": "aws-greengrass-kinesisfirehose",
      "source": "arn:aws:lambda:region:aws:function:aws-greengrass-
kinesisfirehose:version",
      "subject": "kinesisfirehose/message/status",
      "target": "cloud"
    }
  }
}
```

- Ganti *wilayah* dengan Wilayah AWS yang Anda gunakan.
- Ganti *versi* dengan versi fungsi Lambda yang komponen ini jalankan. Untuk menemukan versi fungsi Lambda, Anda harus melihat resep untuk versi komponen ini yang ingin Anda deploy. Buka halaman detail komponen ini di konsol AWS IoT Greengrass tersebut, dan cari pasangan nilai kunci fungsi Lambda. Pasangan kunci-nilai ini berisi nama dan versi fungsi Lambda.

> **Important**
>
> Anda harus memperbarui versi fungsi Lambda pada router langganan warisan setiap kali Anda men-deploy komponen ini. Hal ini memastikan bahwa Anda menggunakan versi fungsi Lambda yang benar untuk versi komponen yang Anda deploy.

Untuk informasi selengkapnya, lihat Buat deployment.

- Komponen Firehose didukung untuk berjalan di VPC. Untuk menerapkan komponen ini di VPC, berikut ini diperlukan.
  - Komponen Firehose harus memiliki konektivitas `firehose.region.amazonaws.com` yang memiliki titik akhir VPC. `com.amazonaws.region.kinesis-firehose`

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Diperlukan	Deskripsi
`firehose.` *`region`*`.amazonaws.com`	443	Ya	Unggah data ke Firehose.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Keras
Peluncur Lambda	^2.0.0	Keras

Dependensi	Versi yang kompatibel	Jenis dependensi
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.8 - 2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.8 dan 2.1.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.7

Tabel berikut mencantumkan dependensi untuk versi 2.0.7 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	^2.0.0	Keras

2.0.6

Tabel berikut mencantumkan dependensi untuk versi 2.0.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.5

Tabel berikut mencantumkan dependensi untuk versi 2.0.5 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.4

Tabel berikut mencantumkan dependensi untuk versi 2.0.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.3

Tabel berikut mencantumkan dependensi untuk versi 2.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Keras
Peluncur Lambda	>=1.0.0	Keras
Runtime Lambda	>=1.0.0	Lunak
Layanan pertukaran Token	>=1.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

> **Note**
>
> Konfigurasi default komponen ini meliputi parameter fungsi Lambda. Kami sarankan Anda mengedit hanya parameter berikut untuk mengonfigurasi komponen ini pada perangkat Anda.

`lambdaParams`

Sebuah objek yang berisi parameter untuk fungsi Lambda komponen ini. Objek ini berisi informasi berikut:

`EnvironmentVariables`

Sebuah objek yang berisi parameter fungsi Lambda ini. Objek ini berisi informasi berikut:

`DEFAULT_DELIVERY_STREAM_ARN`

ARN dari aliran pengiriman Firehose default tempat komponen mengirim data. Anda dapat menimpa aliran tujuan dengan properti `delivery_stream_arn` dalam muatan pesan masukan.

> **Note**
>
> Peran perangkat inti harus memungkinkan tindakan yang diperlukan pada semua aliran pengiriman target. Untuk informasi selengkapnya, lihat Persyaratan.

`PUBLISH_INTERVAL`

(Opsional) Jumlah maksimum detik untuk menunggu sebelum komponen menerbitkan data batch ke Firehose. Untuk mengonfigurasi komponen untuk mempublikasikan metrik saat menerimanya, yang berarti tanpa batching, tentukan `0`.

Nilai ini bisa paling banyak 900 detik.

Default: 10 detik

`DELIVERY_STREAM_QUEUE_SIZE`

(Opsional) Jumlah maksimum rekaman yang disimpan dalam memori sebelum komponen menolak catatan baru untuk aliran pengiriman yang sama.

Nilai ini harus setidaknya berisi 2.000 catatan.

Default: 5.000 catatan

`containerMode`

(Opsional) Mode kontainerisasi untuk komponen ini. Pilih dari salah satu pilihan berikut:

- `NoContainer` – Komponen tersebut tidak berjalan di lingkungan waktu aktif terisolasi.
- `GreengrassContainer`— Komponen berjalan di lingkungan runtime yang terisolasi di dalam AWS IoT Greengrass wadah.

Default: `GreengrassContainer`

`containerParams`

(Opsional) Sebuah objek yang berisi parameter kontainer untuk komponen ini. Komponen menggunakan parameter ini jika Anda menentukan `GreengrassContainer` untuk `containerMode`.

Objek ini berisi informasi berikut:

`memorySize`

(Opsional) Jumlah memori (dalam kilobyte) yang akan dialokasikan ke komponen.

Default ke 64 MB (65.535 KB).

`pubsubTopics`

(Opsional) Sebuah objek yang berisi topik di mana komponen berlangganan untuk menerima pesan. Anda dapat menentukan setiap topik dan apakah komponen berlangganan topik MQTT dari AWS IoT Core atau topik penerbitan/langganan lokal.

Objek ini berisi informasi berikut:

`0` - Ini adalah indeks himpunan sebagai string.

Objek yang berisi informasi berikut:

`type`

(Opsional) Jenis olahpesan publikasikan/berlangganan yang digunakan oleh komponen ini untuk berlangganan pesan. Pilih dari salah satu pilihan berikut:

- `PUB_SUB` — Berlangganan pesan publish/subscribe lokal. Jika Anda memilih opsi ini, topik tidak dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Pesan lokal publikasi/berlangganan.
- `IOT_CORE`— Berlangganan pesan AWS IoT Core MQTT. Jika Anda memilih opsi ini, topik dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim

pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Terbitkan/berlangganan pesan MQTT AWS IoT Core.

Default: PUB_SUB

`topic`

(Opsional) Topik yang menjadi langganan komponen untuk menerima pesan. Jika Anda menentukan `IotCore` untuk `type`, Anda dapat menggunakan wildcard MQTT (+ dan #) dalam topik ini.

Example Contoh: Pembaruan gabungan konfigurasi (mode kontainer)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_DELIVERY_STREAM_ARN": "arn:aws:firehose:us-
west-2:123456789012:deliverystream/mystream"
    }
  },
  "containerMode": "GreengrassContainer"
}
```

Example Contoh: Pembaruan gabungan konfigurasi (tidak ada mode kontainer)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_DELIVERY_STREAM_ARN": "arn:aws:firehose:us-
west-2:123456789012:deliverystream/mystream"
    }
  },
  "containerMode": "NoContainer"
}
```

Data input

Komponen ini menerima konten pengaliran pada topik berikut dan mengirimkan konten ke aliran pengiriman target. Komponen menerima dua jenis data input:

- Data JSON pada topik `kinesisfirehose/message`.

- Data biner pada topik `kinesisfirehose/message/binary/#`.

Topik default untuk data JSON (penerbitan/berlangganan lokal): `kinesisfirehose/message`

Pesan menerima properti berikut. Pesan input harus dalam format JSON.

`request`

Data yang akan dikirim ke aliran pengiriman dan aliran pengiriman target, jika berbeda dari pengaliran default.

Jenis: `object` yang berisi informasi berikut:

`data`

Data yang akan dikirim ke aliran pengiriman.

Tipe: `string`

`delivery_stream_arn`

(Opsional) ARN dari aliran pengiriman Firehose target. Tentukan properti ini untuk menimpa aliran pengiriman default.

Tipe: `string`

`id`

ID acak untuk permintaan. Gunakan properti ini untuk memetakan permintaan inpun untuk respons output. Ketika Anda menentukan properti ini, komponen menetapkan properti `id` di objek respons untuk nilai ini.

Tipe: `string`

Example Contoh input

```
{
  "request": {
    "delivery_stream_arn": "arn:aws:firehose:region:account-id:deliverystream/
stream2-name",
    "data": "Data to send to the delivery stream."
  },
  "id": "request123"
```

```
}
```

Topik default untuk data biner (penerbitan/berlangganan lokal): `kinesisfirehose/message/binary/#`

Gunakan topik ini untuk mengirim pesan yang berisi data biner. Komponen tidak mengurai data biner. Komponen mengalirkan data sebagaimana adanya.

Untuk memetakan permintaan masukan untuk respons output, ganti wildcard # di 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 ke respons, Anda dapat mempublikasikan pesan ke `kinesisfirehose/message/binary/`. Pastikan untuk menyertakan garis miring (`/`).

Data output

Komponen ini menerbitkan tanggapan sebagai data output pada topik MQTT berikut secara default. Anda harus menentukan topik ini sebagai `subject` dalam konfigurasi untuk komponen router langganan warisan. Untuk informasi lebih lanjut tentang cara mempublikasikan pesan ke komponen ini dari komponen kustom Anda, lihat Terbitkan/berlangganan pesan MQTT AWS IoT Core.

Topik default (AWS IoT Core MQTT): `kinesisfirehose/message/status`

Example Contoh Output

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 komponen mendeteksi kesalahan yang dapat dicoba ulang, seperti kesalahan koneksi, ia akan mencoba ulang publikasi di batch berikutnya.

Berkas log lokal

Komponen ini menggunakan file log berikut.

```
/greengrass/v2/logs/aws.greengrass.KinesisFirehose.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* dengan jalur ke folder AWS IoT Greengrass root.

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

Lisensi

Komponen ini 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/MIT License

Komponen ini dirilis menurut Perjanjian Lisensi Perangkat Lunak Greengrass Core.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.5	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.2	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.1	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.0	Fitur baru • Menambahkan dukungan untuk konfigurasi proxy jaringan HTTPS. Lihat informasi yang lebih lengkap di Hubungkan pada port 443 atau melalui proksi jaringan dan Aktifkan perangkat inti untuk mempercayai proxy HTTPS.
2.0.8	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.7	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.6	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.5	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.0.4	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.

Versi	Perubahan
2.0.3	Versi awal.

Lihat juga

- Apa itu Amazon Data Firehose? di Panduan Pengembang Firehose Data Amazon

Peluncur Lambda

Komponen peluncur Lambda (`aws.greengrass.LambdaLauncher`) memulai dan menghentikan AWS Lambda fungsi pada AWS IoT Greengrass perangkat inti. Komponen ini juga mengatur kontainerisasi apa pun dan menjalankan proses sebagai pengguna yang Anda tentukan.

> **Note**
>
> Ketika Anda men-deploy komponen fungsi Lambda ke perangkat inti, deployment juga mencakup komponen ini. Untuk informasi selengkapnya, lihat Jalankan fungsi AWS Lambda.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.0.x

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Anda harus memenuhi persyaratan untuk menjalankan fungsi Lambda. Jika Anda ingin perangkat inti untuk menjalankan fungsi Lambda kontainer, perangkat harus memenuhi persyaratan untuk melakukannya. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.
- Komponen peluncur Lambda didukung untuk berjalan di VPC.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.0.11 – 2.0.13

Tabel berikut mencantumkan dependensi untuk versi 2.0.11 hingga 2.0.13 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Manajer Lambda	>=2.0.0 <2.4.0	Keras

2.0.9 – 2.0.10

Tabel berikut mencantumkan dependensi untuk versi 2.0.9 hingga 2.0.10 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Manajer Lambda	>=2.0.0 <2.3.0	Keras

2.0.4 - 2.0.8

Tabel berikut mencantumkan dependensi untuk versi 2.0.4 hingga 2.0.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Manajer Lambda	>=2.0.0 <2.2.0	Keras

2.0.3

Tabel berikut mencantumkan dependensi untuk versi 2.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Manajer Lambda	>=2.0.3 <2.1.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

Berkas log lokal

Komponen ini menggunakan file log berikut.

```
/greengrass/v2/logs/lambdaFunctionComponentName.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* dengan path ke folder AWS IoT Greengrass root, dan ganti

lambdaFunctionComponentName dengan nama komponen fungsi Lambda yang diluncurkan komponen ini.

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.0.13	Perbaikan bug dan peningkatan Perbaikan bug umum dan perbaikan.
2.0.12	Perbaikan bug dan peningkatan Memperbaiki masalah di mana peluncur Lambda dapat menimbulkan kesalahan jika proses sebelumnya tidak dihentikan dengan benar.
2.0.11	Dukungan untuk manajer Lambda 2.3.0.
2.0.10	Perbaikan bug dan peningkatan • Perbaikan bug umum dan perbaikan.
2.0.9	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.8	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.7	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.6	Peningkatan performa umum dan perbaikan bug.
2.0.4	Perbaikan bug dan peningkatan • Memperbaiki masalah saat komponen tidak meneruskan dengan tepat `AddGroupOwner` ke kontainer fungsi Lambda.
2.0.3	Versi awal.

Manajer Lambda

Komponen manajer Lambda (`aws.greengrass.LambdaManager`) mengelola item kerja dan komunikasi interproses untuk AWS Lambda fungsi yang berjalan pada perangkat inti Greengrass.

> **Note**
>
> Ketika Anda men-deploy komponen fungsi Lambda ke perangkat inti, deployment juga mencakup komponen ini. Untuk informasi selengkapnya, lihat Jalankan fungsi AWS Lambda.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Tipe

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Anda harus memenuhi persyaratan untuk menjalankan fungsi Lambda. Jika Anda ingin perangkat inti untuk menjalankan fungsi Lambda kontainer, perangkat harus memenuhi persyaratan untuk melakukannya. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.
- Komponen manajer Lambda didukung untuk berjalan di VPC.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.3.2 and 2.3.3

Tabel berikut mencantumkan dependensi untuk versi 2.3.2 dan 2.3.3 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak

2.2.10 and 2.3.1

Tabel berikut mencantumkan dependensi untuk versi 2.2.10 dan 2.3.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak

2.2.8 and 2.2.9

Tabel berikut mencantumkan dependensi untuk versi 2.2.8 dan 2.2.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak

2.2.7

Tabel berikut mencantumkan dependensi untuk versi 2.2.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak

2.2.6

Tabel berikut mencantumkan dependensi untuk versi 2.2.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak

2.2.5

Tabel berikut mencantumkan dependensi untuk versi 2.2.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak

2.2.4

Tabel berikut mencantumkan dependensi untuk versi 2.2.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak

2.2.1 - 2.2.3

Tabel berikut mencantumkan dependensi untuk versi 2.2.1 hingga 2.2.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak

2.2.0

Tabel berikut mencantumkan dependensi untuk versi 2.2.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.6.0	Lunak

2.1.3 and 2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 dan 2.1.4 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak

2.0.x

Tabel berikut mencantumkan dependensi untuk versi 2.0.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`logHandlerMode`

> **Note**
>
> Hanya untuk manajer lambda versi 2.3.0+

Digunakan untuk memilih implementasi manajer log Lambda untuk digunakan. Tetapkan nilainya `optimized` untuk menggunakan lebih sedikit utas untuk membaca log lambda.

`getResultTimeoutInSecond`

(Opsional) Jumlah waktu maksimum dalam detik yang dapat dijalankan oleh fungsi Lambda sebelum waktunya habis.

Default: `60`

File log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

```
/greengrass/v2/logs/greengrass.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* dengan jalur ke folder AWS IoT Greengrass root.

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.3.3	Perbaikan bug dan peningkatan • Perbaikan bug umum dan perbaikan.
2.3.2	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.3.1	Perbaikan bug dan peningkatan • Menyesuaikan tingkat log untuk kesalahan tertentu.
2.3.0	Fitur baru • Log handler dioptimalkan untuk mengurangi beban CPU. Gunakan fitur ini dengan mengatur opsi konfigurasi `logHandlerMode` ke`optimized` . Perbaikan bug dan peningkatan • Tidak lagi mencatat stacktrace penuh untuk`WorkQueueFullException` , meningkatkan log dan kinerja. • Menetapkan batas waktu shutdown lambda dari 15 detik hingga 300 detik untuk mencegah batas waktu shutdown. • Memperbaiki masalah di mana lambda sesuai permintaan mungkin gagal memulai ulang setelah mengubah konfigurasi.
2.2.11	Perbaikan bug dan peningkatan • Memperbaiki masalah saat LegacySubscriptionRouter konfigurasi tidak diperbarui saat konfigurasi Lambda berubah.
2.2.10	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.2.9	Perbaikan bug dan peningkatan Memperbaiki masalah di mana nomor port rusak karena jam miring.
2.2.8	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.2.7	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.2.6	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.

Versi	Perubahan
2.2.5	Fitur baru • Menambahkan dukungan untuk wildcard topik MQTT di sumber acara tempat Anda berlangganan pesan penerbitan/berlangganan lokal. Fitur ini membutuhkan v2.6.0 atau yang lebih baru dari komponen inti Greengrass. • Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.2.4	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.2.3	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana beberapa instance fungsi Lambda berbagi cgroup tunggal. Komponen ini menggunakan cgroups untuk mengelola penggunaan sumber daya untuk fungsi Lambda.
2.2.2	Perbaikan bug dan peningkatan • Memperbaiki masalah saat komponen fungsi Lambda yang disematkan dimulai ulang secara tak terduga dalam skenario tertentu.
2.2.1	Perbaikan bug dan peningkatan • Mengubah batasan versi ketergantungan inti Greengrass komponen ini untuk memperbaiki masalah resolusi ketergantungan.
2.2.0	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana fungsi Lambda tidak dapat menulis log setelah restart. • Memperbaiki masalah saat router langganan lama mengirim pesan duplikat saat ada wildcard dalam topik. • Memperbaiki masalah di mana fungsi Lambda yang tidak disematkan tidak dapat menggunakan pustaka komunikasi antarproses Greengrass (IPC) di perpustakaan. AWS IoT Device SDK

Versi	Perubahan
2.1.4	Perbaikan bug dan peningkatan • Memperbaiki masalah yang menyebabkan fungsi Lambda yang menggunakan runtime NodeJS hanya memproses satu pesan. • Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.1.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.1.0	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.
2.0.3	Versi awal.

Runtime Lambda

Komponen waktu aktif Lambda (`aws.greengrass.LambdaRuntimes`) menyediakan waktu aktif yang digunakan oleh perangkat inti Greengrass untuk menjalankan fungsi AWS Lambda.

> **Note**
>
> Ketika Anda men-deploy komponen fungsi Lambda ke perangkat inti, deployment juga mencakup komponen ini. Untuk informasi selengkapnya, lihat Jalankan fungsi AWS Lambda.

Topik

- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.0.x

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Anda harus memenuhi persyaratan untuk menjalankan fungsi Lambda. Jika Anda ingin perangkat inti untuk menjalankan fungsi Lambda kontainer, perangkat harus memenuhi persyaratan untuk melakukannya. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.
- Komponen runtime Lambda didukung untuk berjalan di VPC.

Dependensi

Komponen ini tidak memiliki dependensi apa pun.

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

Berkas log lokal

Komponen ini tidak mengeluarkan log.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.0.8	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.7	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.6	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.5	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.0.4	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.
2.0.3	Versi awal.

Router langganan warisan

Router langganan warisan (`aws.greengrass.LegacySubscriptionRouter`) mengelola langganan pada perangkat inti Greengrass. Langganan adalah fitur dari V1 AWS IoT Greengrass yang menentukan topik yang dapat digunakan Lambda fungsi untuk olah pesan MQTT pada perangkat inti. Untuk informasi selengkapnya, lihat Langganan terkelola dalam alur kerja olahpesan MQTT di Panduan Developer V1 AWS IoT Greengrass.

Anda dapat menggunakan komponen ini untuk mengaktifkan langganan untuk komponen konektor dan komponen fungsi Lambda yang menggunakan SDK Inti AWS IoT Greengrass.

> **Note**
>
> Komponen router langganan lama hanya diperlukan jika fungsi Lambda Anda menggunakan fungsi `publish()` di Core AWS IoT Greengrass SDK. Jika Anda memperbarui kode fungsi Lambda Anda untuk menggunakan antarmuka komunikasi antarproses (IPC) di AWS IoT Device SDK V2, Anda tidak perlu menggunakan komponen router langganan lama. Untuk informasi lebih lanjut, lihat layanan komunikasi antar proses berikut ini:
>
> - Pesan lokal publikasi/berlangganan

- Terbitkan/berlangganan pesan MQTT AWS IoT Core

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Router langganan lama didukung untuk berjalan di VPC.

Dependensi

Saat Anda men-deploy komponen, AWS IoT Greengrass juga men-deploy versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.11

Tabel berikut mencantumkan dependensi untuk versi 2.1.11 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak

2.1.10

Tabel berikut mencantumkan dependensi untuk versi 2.1.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak

2.1.9

Tabel berikut mencantumkan dependensi untuk versi 2.1.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak

2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak

2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak

2.0.3

Tabel berikut mencantumkan dependensi untuk versi 2.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

v2.1.x

`subscriptions`

(Opsional) Langganan yang akan diaktifkan pada perangkat inti. Ini adalah objek, di mana setiap kunci adalah ID unik, dan setiap nilai adalah objek yang mendefinisikan langganan untuk konektor itu. Anda harus mengonfigurasi langganan ketika Anda men-deploy komponen konektor V1 atau fungsi Lambda yang menggunakan SDK Inti AWS IoT Greengrass.

Setiap objek langganan berisi informasi berikut.

`id`

ID unik langganan ini. ID ini harus sesuai dengan kunci untuk objek langganan ini.

`source`

Fungsi Lambda yang menggunakan SDK Inti AWS IoT Greengrass untuk mempublikasikan pesan MQTT pada topik yang Anda tentukan di `subject`. Tentukan satu dari yang berikut ini:

- Nama komponen fungsi Lambda pada perangkat inti. Tentukan nama komponen dengan prefiks `component:`, seperti **`component:com.example.HelloWorldLambda`**.
- Amazon Resource Name (ARN) dari fungsi Lambda pada perangkat inti.

> **Important**
>
> Jika versi fungsi Lambda berubah, Anda harus mengonfigurasi langganan dengan versi baru fungsi. Jika tidak, komponen ini tidak akan mengarahkan pesan hingga versi itu cocok dengan langganan tersebut.

Anda harus menentukan Nama Sumber Daya Amazon (ARN) yang menyertakan versi fungsi yang akan diimpor. Anda tidak dapat menggunakan alias versi seperti `$LATEST`.

Untuk menggunakan langganan untuk komponen konektor V1, tentukan nama komponen atau ARN dari komponen konektor fungsi Lambda.

`subject`

Topik MQTT atau filter topik di mana sumber dan target dapat mempublikasikan dan menerima pesan. Nilai ini mendukung wildcard topik + dan #.

`target`

Target yang menerima pesan MQTT pada topik yang Anda tentukan di `subject`. Langganan menentukan bahwa fungsi `source` menerbitkan pesan MQTT ke AWS IoT Core atau ke fungsi Lambda pada perangkat inti. Tentukan satu dari yang berikut ini:

- `cloud`. Fungsi `source` menerbitkan pesan MQTT untuk AWS IoT Core.
- Nama komponen fungsi Lambda pada perangkat inti. Tentukan nama komponen dengan prefiks `component:`, seperti **`component:com.example.HelloWorldLambda`**.
- Amazon Resource Name (ARN) dari fungsi Lambda pada perangkat inti.

Important

Jika versi fungsi Lambda berubah, Anda harus mengonfigurasi langganan dengan versi baru fungsi. Jika tidak, komponen ini tidak akan mengarahkan pesan hingga versi itu cocok dengan langganan tersebut.
Anda harus menentukan Nama Sumber Daya Amazon (ARN) yang menyertakan versi fungsi yang akan diimpor. Anda tidak dapat menggunakan alias versi seperti `$LATEST`.

Default: Tidak ada langganan

Example Contoh pembaruan konfigurasi (mendefinisikan langganan AWS IoT Core)

Contoh berikut menetapkan bahwa komponen fungsi Lambda `com.example.HelloWorldLambda` menerbitkan pesan MQTT untuk AWS IoT Core pada topik `hello/world`.

```
{
  "subscriptions": {
    "Greengrass_HelloWorld_to_cloud": {
      "id": "Greengrass_HelloWorld_to_cloud",
      "source": "component:com.example.HelloWorldLambda",
      "subject": "hello/world",
      "target": "cloud"
    }
  }
}
```

Example Contoh pembaruan konfigurasi (mendefinisikan langganan ke fungsi Lambda)

Contoh berikut menetapkan bahwa komponen fungsi Lambda `com.example.HelloWorldLambda` menerbitkan pesan MQTT untuk komponen fungsi Lambda `com.example.MessageRelay` pada topik `hello/world`.

```
{
  "subscriptions": {
    "Greengrass_HelloWorld_to_MessageRelay": {
      "id": "Greengrass_HelloWorld_to_MessageRelay",
      "source": "component:com.example.HelloWorldLambda",
      "subject": "hello/world",
      "target": "component:com.example.MessageRelay"
    }
  }
}
```

v2.0.x

`subscriptions`

(Opsional) Langganan yang akan diaktifkan pada perangkat inti. Ini adalah objek, di mana setiap kunci adalah ID unik, dan setiap nilai adalah objek yang mendefinisikan langganan untuk konektor itu. Anda harus mengonfigurasi langganan ketika Anda men-deploy komponen konektor V1 atau fungsi Lambda yang menggunakan SDK Inti AWS IoT Greengrass.

Setiap objek langganan berisi informasi berikut.

`id`

ID unik langganan ini. ID ini harus sesuai dengan kunci untuk objek langganan ini.

`source`

Fungsi Lambda yang menggunakan SDK Inti AWS IoT Greengrass untuk mempublikasikan pesan MQTT pada topik yang Anda tentukan di `subject`. Tentukan hal berikut:

- Amazon Resource Name (ARN) dari fungsi Lambda pada perangkat inti.

> **Important**
>
> Jika versi fungsi Lambda berubah, Anda harus mengonfigurasi langganan dengan versi baru fungsi. Jika tidak, komponen ini tidak akan mengarahkan pesan hingga versi itu cocok dengan langganan tersebut.
> Anda harus menentukan Nama Sumber Daya Amazon (ARN) yang menyertakan versi fungsi yang akan diimpor. Anda tidak dapat menggunakan alias versi seperti `$LATEST`.

Untuk menggunakan langganan untuk komponen konektor V1, tentukan ARN dari komponen konektor fungsi Lambda.

`subject`

Topik MQTT atau filter topik di mana sumber dan target dapat mempublikasikan dan menerima pesan. Nilai ini mendukung wildcard topik + dan #.

`target`

Target yang menerima pesan MQTT pada topik yang Anda tentukan di `subject`. Langganan menentukan bahwa fungsi `source` menerbitkan pesan MQTT ke AWS IoT Core atau ke fungsi Lambda pada perangkat inti. Tentukan satu dari yang berikut ini:

- `cloud`. Fungsi `source` menerbitkan pesan MQTT untuk AWS IoT Core.
- Amazon Resource Name (ARN) dari fungsi Lambda pada perangkat inti.

> **Important**
>
> Jika versi fungsi Lambda berubah, Anda harus mengonfigurasi langganan dengan versi baru fungsi. Jika tidak, komponen ini tidak akan mengarahkan pesan hingga versi itu cocok dengan langganan tersebut.
> Anda harus menentukan Nama Sumber Daya Amazon (ARN) yang menyertakan versi fungsi yang akan diimpor. Anda tidak dapat menggunakan alias versi seperti `$LATEST`.

Default: Tidak ada langganan

Example Contoh pembaruan konfigurasi (mendefinisikan langganan AWS IoT Core)

Contoh berikut menetapkan bahwa fungsi `Greengrass_HelloWorld` menerbitkan pesan MQTT ke AWS IoT Core pada topik `hello/world`.

```
"subscriptions": {
  "Greengrass_HelloWorld_to_cloud": {
    "id": "Greengrass_HelloWorld_to_cloud",
    "source": "arn:aws:lambda:us-
west-2:123456789012:function:Greengrass_HelloWorld:5",
    "subject": "hello/world",
    "target": "cloud"
  }
}
```

Example Contoh pembaruan konfigurasi (mendefinisikan langganan ke fungsi Lambda)

Contoh berikut menetapkan bahwa fungsi `Greengrass_HelloWorld` menerbitkan pesan MQTT ke `Greengrass_MessageRelay` pada topik `hello/world`.

```
"subscriptions": {
  "Greengrass_HelloWorld_to_MessageRelay": {
    "id": "Greengrass_HelloWorld_to_MessageRelay",
    "source": "arn:aws:lambda:us-
west-2:123456789012:function:Greengrass_HelloWorld:5",
    "subject": "hello/world",
    "target": "arn:aws:lambda:us-
west-2:123456789012:function:Greengrass_MessageRelay:5"
  }
}
```

Berkas log lokal

Komponen ini tidak mengeluarkan log.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.11	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.10	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.9	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.5	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.1.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.1.0	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk menentukan nama komponen dan bukan ARN untuk `source` dan `target`. Jika Anda menentukan nama komponen untuk suatu langganan, Anda tidak perlu mengonfigurasi ulang langganan setiap kali versi fungsi Lambda berubah.
2.0.3	Versi awal.

Konsol debug lokal

Komponen konsol debug lokal (`aws.greengrass.LocalDebugConsole`) menyediakan dasbor lokal yang menampilkan informasi tentang perangkat AWS IoT Greengrass inti Anda dan komponennya. Anda dapat menggunakan dasbor ini untuk men-debug perangkat inti Anda dan mengelola komponen lokal.

> **Important**
>
> Kami menyarankan Anda menggunakan komponen ini hanya di lingkungan pengembangan, bukan lingkungan produksi. Komponen ini menyediakan akses ke informasi dan operasi yang biasanya tidak Anda perlukan di lingkungan produksi. Ikuti prinsip hak istimewa paling sedikit dengan menerapkan komponen ini hanya ke perangkat inti di mana Anda membutuhkannya.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Anda menggunakan nama pengguna dan kata sandi untuk masuk ke dasbor. Nama pengguna, yaitu `debug`, disediakan untuk Anda. Anda harus menggunakan AWS IoT Greengrass CLI untuk membuat kata sandi sementara yang mengautentikasi Anda dengan dasbor pada perangkat inti. Anda harus dapat menggunakan AWS IoT Greengrass CLI untuk menggunakan konsol debug lokal. Untuk informasi lebih lanjut, lihat Persyaratan Greengrass CLI. Untuk informasi lebih lanjut tentang cara membuat kata sandi dan masuk, lihat Penggunaan komponen konsol debug lokal.
- Komponen konsol debug lokal didukung untuk berjalan di VPC.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.4.1 – 2.4.2

Tabel berikut mencantumkan dependensi untuk versi 2.4.1 hingga 2.4.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.10.0 <2.13.0	Keras
CLI Greengrass	>=2.10.0 <2.13.0	Keras

2.4.0

Tabel berikut mencantumkan dependensi untuk versi 2.4.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.10.0 <2.12.0	Keras
CLI Greengrass	>=2.10.0 <2.12.0	Keras

2.3.0 and 2.3.1

Tabel berikut mencantumkan dependensi untuk versi 2.3.0 dan 2.3.1 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.10.0 <2.12.0	Keras
CLI Greengrass	>=2.10.0 <2.12.0	Keras

2.2.9

Tabel berikut mencantumkan dependensi untuk versi 2.2.9 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.12.0	Keras
CLI Greengrass	>=2.1.0 <2.12.0	Keras

2.2.8

Tabel berikut mencantumkan dependensi untuk versi 2.2.8 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.11.0	Keras
CLI Greengrass	>=2.1.0 <2.11.0	Keras

2.2.7

Tabel berikut mencantumkan dependensi untuk versi 2.2.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.10.0	Keras
CLI Greengrass	>=2.1.0 <2.10.0	Keras

2.2.6

Tabel berikut mencantumkan dependensi untuk versi 2.2.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.9.0	Keras
CLI Greengrass	>=2.1.0 <2.9.0	Keras

2.2.5

Tabel berikut mencantumkan dependensi untuk versi 2.2.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.8.0	Keras

Dependensi	Versi yang kompatibel	Jenis dependensi
CLI Greengrass	>=2.1.0 <2.8.0	Keras

2.2.4

Tabel berikut mencantumkan dependensi untuk versi 2.2.4 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.7.0	Keras
CLI Greengrass	>=2.1.0 <2.7.0	Keras

2.2.3

Tabel berikut mencantumkan dependensi untuk versi 2.2.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.6.0	Keras
CLI Greengrass	>=2.1.0 <2.6.0	Keras

2.2.2

Tabel berikut mencantumkan dependensi untuk versi 2.2.2 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.5.0	Keras
CLI Greengrass	>=2.1.0 <2.5.0	Keras

2.2.1

Tabel berikut mencantumkan dependensi untuk versi 2.2.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.4.0	Keras
CLI Greengrass	>=2.1.0 <2.4.0	Keras

2.2.0

Tabel berikut mencantumkan dependensi untuk versi 2.2.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.3.0	Keras
CLI Greengrass	>=2.1.0 <2.3.0	Keras

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.2.0	Keras
CLI Greengrass	>=2.1.0 <2.2.0	Keras

2.0.x

Tabel berikut mencantumkan dependensi untuk versi 2.0.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Lunak
CLI Greengrass	>=2.0.3 <2.1.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

v2.1.x - v2.4.x

`httpsEnabled`

(Opsional) Anda dapat mengaktifkan HTTPS komunikasi untuk konsol debug lokal. Jika Anda mengaktifkan HTTPS komunikasi, konsol debug lokal akan membuat sertifikat yang ditandatangani sendiri. Peramban web menampilkan peringatan keamanan untuk situs web yang menggunakan sertifikat yang ditandatangani sendiri, sehingga Anda harus memverifikasi sertifikat secara manual. Kemudian, Anda dapat melewati peringatan ini. Untuk informasi selengkapnya, lihat Penggunaan.

Default: `true`

`port`

(Opsional) Port tempat untuk menyediakan konsol debug lokal.

Default: `1441`

`websocketPort`

(Opsional) Port websocket yang akan digunakan untuk konsol debug lokal.

Default: `1442`

`bindHostname`

(Opsional) Nama host yang akan digunakan untuk konsol debug lokal.

Jika Anda menjalankan perangkat lunak AWS IoT Greengrass Core dalam wadah Docker, atur parameter ini`0.0.0.0`, sehingga Anda dapat membuka konsol debug lokal di luar wadah Docker.

Default: `localhost`

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk membuka konsol debug lokal pada port non-default dan menonaktifkan HTTPS.

```
{
  "httpsEnabled": false,
  "port": "10441",
  "websocketPort": "10442"
}
```

v2.0.x

`port`

(Opsional) Port tempat untuk menyediakan konsol debug lokal.

Default: `1441`

`websocketPort`

(Opsional) Port websocket yang akan digunakan untuk konsol debug lokal.

Default: `1442`

`bindHostname`

(Opsional) Nama host yang akan digunakan untuk konsol debug lokal.

Jika Anda menjalankan perangkat lunak AWS IoT Greengrass Core dalam wadah Docker, atur parameter ini`0.0.0.0`, sehingga Anda dapat membuka konsol debug lokal di luar wadah Docker.

Default: `localhost`

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk membuka konsol debug lokal pada port non-default.

```
{
  "port": "10441",
  "websocketPort": "10442"
}
```

Penggunaan

Untuk menggunakan konsol debug lokal, buat sesi dari Greengrass CLI. Ketika Anda membuat sesi, Greengrass CLI akan menyediakan nama pengguna dan kata sandi sementara yang dapat Anda gunakan untuk masuk ke konsol debug lokal.

Ikuti petunjuk ini untuk membuka konsol debug lokal pada perangkat inti atau komputer pengembangan Anda.

v2.1.x - v2.4.x

Dalam versi 2.1.0 dan kemudian, konsol debug lokal menggunakan HTTPS secara default. Ketika HTTPS diaktifkan, konsol debug lokal membuat sertifikat yang ditandatangani sendiri untuk mengamankan sambungan. Peramban web Anda akan menunjukkan peringatan keamanan ketika Anda membuka konsol debug lokal karena sertifikat yang ditandatangani sendiri ini. Ketika Anda membuat sesi dengan Greengrass CLI, output akan mencakup sidik jari sertifikat, sehingga Anda dapat memverifikasi bahwa sertifikat itu sah dan sambungan aman.

Anda dapat menonaktifkan HTTPS. Untuk informasi lebih lanjut, lihat konfigurasi konsol debug lokal.

Untuk membuka konsol debug lokal

1. (Opsional) Untuk melihat konsol debug lokal pada komputer pengembangan Anda, Anda dapat meneruskan port konsol melalui SSH. Namun, Anda harus mengaktifkan opsi `AllowTcpForwarding` di file konfigurasi SSH perangkat inti Anda. Opsi ini diatur secara default. Jalankan perintah berikut pada komputer pengembangan Anda untuk melihat dasbor di `localhost:1441` pada komputer pengembangan Anda.

   ```
   ssh -L 1441:localhost:1441 -L 1442:localhost:1442 username@core-device-ip-address
   ```

   > **Note**
   >
   > Anda dapat mengubah port default dari `1441` dan `1442`. Untuk informasi lebih lanjut, lihat konfigurasi konsol debug lokal.

2. Buat sesi untuk menggunakan konsol debug lokal. Ketika Anda membuat sesi, Anda membuat kata sandi yang Anda gunakan untuk mengautentikasi. Konsol debug lokal

memerlukan kata sandi untuk meningkatkan keamanan, karena Anda dapat menggunakan komponen ini untuk melihat informasi penting dan melakukan operasi pada perangkat inti. Konsol debug lokal juga menciptakan sertifikat untuk mengamankan sambungan jika Anda mengaktifkan HTTPS dalam konfigurasi komponen. HTTPS tidak diaktifkan secara default.

Gunakan AWS IoT Greengrass CLI untuk membuat sesi. Perintah ini menghasilkan kata sandi 43 karakter acak yang akan kedaluwarsa setelah 8 jam. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass V2 root.

Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli get-debug-password
```

Windows

```
C:\greengrass\v2\bin\greengrass-cli get-debug-password
```

Perintah output terlihat seperti contoh berikut jika Anda telah mengonfigurasi konsol debug lokal untuk menggunakan HTTPS. Anda menggunakan sidik jari sertifikat untuk memverifikasi bahwa sambungan aman ketika Anda membuka konsol debug lokal.

```
Username: debug
Password: bEDp3MOHdj8ou2w5de_sCBI2XAaguy3a8XxREXAMPLE
Password expires at: 2021-04-01T17:01:43.921999931-07:00
The local debug console is configured to use TLS security. The certificate is
 self-signed so you will need to bypass your web browser's security warnings to
 open the console.
Before you bypass the security warning, verify that the certificate fingerprint
 matches the following fingerprints.
SHA-256: 15 0B 2C E2 54 8B 22 DE 08 46 54 8A B1 2B 25 DE FB 02 7D 01 4E 4A 56 67
 96 DA A6 CC B1 D2 C4 1B
SHA-1: BC 3E 16 04 D3 80 70 DA E0 47 25 F9 90 FA D6 02 80 3E B5 C1
```

Komponen tampilan debug menciptakan sesi yang berlangsung selama 8 jam. Setelah itu, Anda harus menghasilkan kata sandi baru untuk melihat konsol debug lokal lagi.

3. Buka dan masuk ke dasbor. Lihat dasbor pada perangkat inti Greengrass Anda, atau pada komputer pengembangan Anda jika Anda meneruskan port melalui SSH. Lakukan salah satu hal berikut:

- Jika Anda mengaktifkan HTTPS di konsol debug lokal, yang merupakan pengaturan default, lakukan hal berikut:

  a. Buka `https://localhost:1441` pada perangkat inti Greengrass Anda, atau pada komputer pengembangan Anda jika Anda meneruskan port melalui SSH.

     Peramban Anda mungkin akan menampilkan peringatan keamanan tentang sertifikat keamanan yang tidak valid.

  b. Jika peramban Anda menampilkan peringatan keamanan, verifikasi bahwa sertifikat itu sah dan lewati peringatan keamanan. Lakukan hal-hal berikut:

     i. Temukan sidik jari SHA-256 atau SHA-1 untuk sertifikat itu, dan verifikasi bahwa ia cocok dengan sidik jari SHA-256 atau SHA-1 yang dicetak oleh perintah `get-debug-password` sebelumnya. Peramban Anda mungkin akan menyediakan satu atau kedua sidik jari itu. Lihat dokumentasi peramban Anda untuk melihat sertifikat dan sidik jarinya. Di beberapa peramban, sidik jari sertifikat disebut sidik jari.

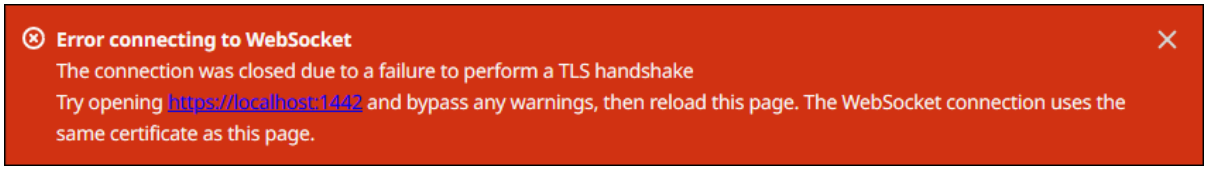
     > **Note**
     >
     > Jika sidik jari sertifikat tidak cocok, buka Step 2 untuk membuat sesi baru. Jika sidik jari sertifikat masih tidak cocok, koneksi Anda mungkin tidak aman.

     ii. Jika sidik jari sertifikat cocok, lewati peringatan keamanan peramban Anda untuk membuka konsol debug lokal. Lihat dokumentasi peramban Anda untuk melewati peringatan keamanan peramban.

  c. Masuk ke situs web dengan menggunakan nama pengguna dan kata sandi yang dicetak oleh perintah `get-debug-password` sebelumnya.

     Konsol debug lokal terbuka.

  d. Jika konsol debug lokal menunjukkan kesalahan yang mengatakan tidak dapat terhubung ke konsol WebSocket karena jabat tangan TLS gagal, Anda harus melewati peringatan keamanan yang ditandatangani sendiri untuk URL tersebut. WebSocket



Lakukan hal-hal berikut:

i. Buka `https://localhost:1442` di peramban yang sama di mana Anda membuka konsol debug lokal.

ii. Verifikasi sertifikat dan lewati peringatan keamanan.

Peramban Anda mungkin akan menampilkan halaman HTTP 404 setelah Anda melewati peringatan.

iii. Buka `https://localhost:1441` lagi.

Konsol debug lokal menunjukkan informasi tentang perangkat inti.

- Jika Anda menonaktifkan HTTPS di konsol debug lokal, lakukan hal berikut:

a. Buka `http://localhost:1441` pada perangkat inti, atau buka pada komputer pengembangan Anda jika Anda meneruskan port melalui SSH.

b. Masuk ke situs web dengan menggunakan nama pengguna dan kata sandi yang dicetak oleh perintah `get-debug-password` sebelumnya.

Konsol debug lokal terbuka.

v2.0.x

Untuk membuka konsol debug lokal

1. (Opsional) Untuk melihat konsol debug lokal pada komputer pengembangan Anda, Anda dapat meneruskan port konsol melalui SSH. Namun, Anda harus mengaktifkan opsi `AllowTcpForwarding` di file konfigurasi SSH perangkat inti Anda. Opsi ini diatur secara default. Jalankan perintah berikut pada komputer pengembangan Anda untuk melihat dasbor di `localhost:1441` pada komputer pengembangan Anda.

```
ssh -L 1441:localhost:1441 -L 1442:localhost:1442 username@core-device-ip-address
```

> **Note**
>
> Anda dapat mengubah port default dari 1441 dan 1442. Untuk informasi lebih lanjut, lihat konfigurasi konsol debug lokal.

2. Buat sesi untuk menggunakan konsol debug lokal. Ketika Anda membuat sesi, Anda membuat kata sandi yang Anda gunakan untuk mengautentikasi. Konsol debug lokal memerlukan kata sandi untuk meningkatkan keamanan, karena Anda dapat menggunakan komponen ini untuk melihat informasi penting dan melakukan operasi pada perangkat inti.

   Gunakan AWS IoT Greengrass CLI untuk membuat sesi. Perintah ini menghasilkan kata sandi 43 karakter acak yang akan kedaluwarsa setelah 8 jam. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass V2 root.

   Linux or Unix

   ```
   sudo /greengrass/v2/bin/greengrass-cli get-debug-password
   ```

   Windows

   ```
   C:\greengrass\v2\bin\greengrass-cli get-debug-password
   ```

   Perintah output terlihat seperti contoh berikut.

   ```
   Username: debug
   Password: bEDp3MOHdj8ou2w5de_sCBI2XAaguy3a8XxREXAMPLE
   Password will expire at: 2021-04-01T17:01:43.921999931-07:00
   ```

   Komponen tampilan debug menciptakan sesi yang berlangsung selama 4 jam, dan kemudian Anda harus menghasilkan kata sandi baru untuk melihat konsol debug lokal lagi.

3. Buka `http://localhost:1441` pada perangkat inti, atau buka pada komputer pengembangan Anda jika Anda meneruskan port melalui SSH.

4. Masuk ke situs web dengan menggunakan nama pengguna dan kata sandi yang dicetak oleh perintah `get-debug-password` sebelumnya.

   Konsol debug lokal terbuka.

File log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.4.2	Perbaikan bug dan peningkatan • Perbaikan bug umum dan perbaikan.
2.4.1	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.

Versi	Perubahan
2.4.0	Fitur baru • Menambahkan konsol debugging manajer aliran.
2.3.1	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.3.0	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis. Fitur baru • Termasuk PubSub dan klien AWS IoT Core debug MQTT.
2.2.7	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.2.6	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.2.5	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.2.4	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.2.3	Perbaikan bug dan peningkatan • Memperbaiki masalah yang mencegah startup ketika komponen tidak dapat mendekripsi keystore yang menyimpan kunci pribadi SSL. • Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.2.2	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.2.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.2.0	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.1.0	Fitur baru • Menggunakan HTTPS untuk mengamankan koneksi Anda ke konsol debug lokal. HTTPS tidak diaktifkan secara default. Perbaikan bug dan peningkatan • Anda dapat mengabaikan pesan flashbar di editor konfigurasi.
2.0.3	Versi awal.

Manajer log

Komponen pengelola log (`aws.greengrass.LogManager`) mengunggah log dari perangkat AWS IoT Greengrass inti ke Amazon CloudWatch Logs. Anda dapat meng-upload log dari inti Greengrass, komponen Greengrass lainnya, dan aplikasi dan layanan lain yang bukan komponen Greengrass. Untuk informasi selengkapnya tentang cara memantau CloudWatch log di Log dan pada sistem file lokal, lihatMemantau AWS IoT Greengrass log.

Pertimbangan berikut berlaku saat Anda menggunakan komponen pengelola log untuk menulis ke CloudWatch Log:

- Penundaan log

  > **Note**
  >
  > Kami menyarankan Anda meningkatkan ke pengelola log versi 2.3.0 yang mengurangi penundaan log untuk file log yang diputar dan aktif. Saat Anda meningkatkan ke pengelola log 2.3.0, kami sarankan Anda juga meningkatkan ke Greengrass nucleus 2.9.1.

  Komponen pengelola log versi 2.2.8 (dan sebelumnya) memproses dan mengunggah log hanya dari file log yang diputar. Secara default, perangkat lunak AWS IoT Greengrass Core memutar file log setiap jam atau setelah 1.024 KB. Akibatnya, komponen pengelola log mengunggah log hanya setelah perangkat lunak AWS IoT Greengrass Core atau komponen Greengrass menulis log senilai lebih dari 1.024 KB. Anda dapat mengonfigurasi batas ukuran file log yang lebih rendah untuk menyebabkan file log diputar lebih sering. Hal ini menyebabkan komponen pengelola log mengunggah CloudWatch log ke Log lebih sering.

  Komponen pengelola log versi 2.3.0 (dan yang lebih baru) memproses dan mengunggah semua log. Saat Anda menulis log baru, pengelola log versi 2.3.0 (dan yang lebih baru) memproses dan langsung mengunggah file log aktif itu alih-alih menunggu untuk diputar. Ini berarti Anda dapat melihat log baru dalam 5 menit atau kurang.

  Komponen pengelola log mengunggah log baru secara berkala. Secara default, komponen pengelola log mengunggah log baru setiap 5 menit. Anda dapat mengonfigurasi interval unggahan yang lebih rendah, sehingga komponen pengelola log mengunggah CloudWatch log ke Log lebih sering dengan mengonfigurasi log. `periodicUploadIntervalSec` Untuk informasi selengkapnya tentang cara mengonfigurasi interval periodik ini, lihat Konfigurasi.

Log dapat diunggah dalam waktu dekat dari sistem file Greengrass yang sama. Jika Anda perlu mengamati log secara real time, pertimbangkan untuk menggunakan log sistem file.

> **Note**
>
> Jika Anda menggunakan sistem file yang berbeda untuk menulis log, pengelola log kembali ke perilaku di komponen pengelola log versi 2.2.8 dan yang lebih lama. Untuk informasi tentang mengakses log sistem file, lihat Mengakses log sistem file.

- Kemiringan jam

  Komponen pengelola log menggunakan proses penandatanganan Signature Version 4 standar untuk membuat permintaan API ke CloudWatch Log. Jika waktu sistem pada perangkat inti tidak sinkron lebih dari 15 menit, maka CloudWatch Log menolak permintaan. Untuk informasi selengkapnya, lihat Proses penandatanganan Versi Tanda Tangan 4 di bagian Referensi Umum AWS.

Untuk informasi tentang grup log dan pengaliran log yang padanya komponen ini meng-upload log, lihat Penggunaan.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Peran perangkat Greengrass harus mengizinkan tindakan `logs:CreateLogGroup`, `logs:CreateLogStream`, `logs:PutLogEvents`, dan `logs:DescribeLogStreams`, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
```

```
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

> **Note**
>
> Peran perangkat Greengrass yang Anda buat ketika Anda menginstal perangkat lunak inti AWS IoT Greengrass mencakup izin dalam contoh kebijakan ini secara default.

Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk Log CloudWatch di Panduan Pengguna Log Amazon CloudWatch .

- Komponen pengelola log didukung untuk berjalan di VPC. Untuk menerapkan komponen ini di VPC, berikut ini diperlukan.
  - Komponen pengelola log harus memiliki konektivitas `logs.region.amazonaws.com` yang memiliki titik akhir VPC. `com.amazonaws.us-east-1.logs`

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Diperlukan	Deskripsi
`logs.region.amazonaws.com`	443	Tidak	Diperlukan jika Anda menulis log ke CloudWatch Log.

Dependensi

Saat Anda men-deploy komponen, AWS IoT Greengrass juga men-deploy versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.3.7

Tabel berikut mencantumkan dependensi untuk versi 2.3.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.13.0	Lunak

2.3.5 and 2.3.6

Tabel berikut mencantumkan dependensi untuk versi 2.3.5 dan 2.3.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.12.0	Lunak

2.3.3 – 2.3.4

Tabel berikut mencantumkan dependensi untuk versi 2.3.3 hingga 2.3.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.11.0	Lunak

2.2.8 – 2.3.2

Tabel berikut mencantumkan dependensi untuk versi 2.2.8 hingga 2.3.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.10.0	Lunak

2.2.7

Tabel berikut mencantumkan dependensi untuk versi 2.2.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.9.0	Lunak

2.2.6

Tabel berikut mencantumkan dependensi untuk versi 2.2.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.8.0	Lunak

2.2.5

Tabel berikut mencantumkan dependensi untuk versi 2.2.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.7.0	Lunak

2.2.1 - 2.2.4

Tabel berikut mencantumkan dependensi untuk versi 2.2.1 - 2.2.4 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.6.0	Lunak

2.1.3 and 2.2.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 dan 2.2.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.5.0	Lunak

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.4.0	Lunak

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.3.0	Lunak

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.1.0 <2.2.0	Lunak

2.0.x

Tabel berikut mencantumkan dependensi untuk versi 2.0.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

v2.3.6 – v2.3.7

`logsUploaderConfiguration`

(Opsional) Konfigurasi untuk log yang di-upload oleh komponen manajer log. Objek ini berisi informasi berikut:

`systemLogsConfiguration`

(Opsional) Konfigurasi untuk log sistem perangkat lunak AWS IoT Greengrass Core, yang mencakup log dari inti Greengrass dan komponen plugin. Tentukan konfigurasi ini untuk mengaktifkan komponen pengelola log mengelola log sistem. Objek ini berisi informasi berikut:

`uploadToCloudWatch`

(Opsional) Anda dapat mengunggah log sistem ke CloudWatch Log.

Default: `false`

`minimumLogLevel`

(Opsional) Tingkat minimum pesan log untuk diunggah. Level minimum ini hanya berlaku jika Anda mengonfigurasi komponen inti Greengrass untuk mengeluarkan log format JSON. Untuk mengaktifkan log format JSON, `JSON` tentukan parameter format logging (`logging.format`).

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`
- `INFO`

- `WARN`
- `ERROR`

Default: `INFO`

`diskSpaceLimit`

(Opsional) Ukuran total maksimum file log sistem Greengrass, di unit yang Anda tentukan. `diskSpaceLimitUnit` Setelah ukuran total file log sistem Greengrass melebihi ukuran total maksimum ini, AWS IoT Greengrass perangkat lunak Core menghapus file log sistem Greengrass tertua.

Parameter ini setara dengan parameter batas ukuran log (`totalLogsSizeKB`) dari komponen inti Greengrass. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log sistem Greengrass total maksimum.

`diskSpaceLimitUnit`

(Opsional) Unit untuk `diskSpaceLimit`. Pilih dari salah satu pilihan berikut:

- `KB` – kilobyte
- `MB` – megabyte
- `GB` – gigabyte

Default: `KB`

`deleteLogFileAfterCloudUpload`

(Opsional) Anda dapat menghapus file log setelah komponen pengelola log mengunggah CloudWatch log ke Log.

Default: `false`

`componentLogsConfigurationMap`

(Opsional) Peta konfigurasi log untuk komponen pada perangkat inti. Setiap `componentName` objek dalam peta ini mendefinisikan konfigurasi log untuk komponen atau aplikasi. Komponen pengelola log mengunggah log komponen ini ke CloudWatch Log.

> **Important**
>
> Kami sangat menyarankan menggunakan satu kunci konfigurasi per komponen. Anda hanya boleh menargetkan sekelompok file yang hanya memiliki satu file log

yang secara aktif ditulis saat menggunakan file`logFileRegex`. Tidak mengikuti rekomendasi ini dapat menyebabkan log duplikat diunggah. CloudWatch Jika Anda menargetkan beberapa file log aktif dengan satu regex, kami sarankan Anda meningkatkan ke pengelola log v2.3.1 atau yang lebih baru dan pertimbangkan untuk mengubah konfigurasi Anda menggunakan konfigurasi contoh.

> **Note**
>
> Jika Anda memutakhirkan dari versi pengelola log lebih awal dari v2.2.0, Anda dapat terus menggunakan `componentLogsConfiguration` daftar alih-alih. `componentLogsConfigurationMap` Namun, kami sangat menyarankan Anda menggunakan format peta sehingga Anda dapat menggunakan menggabungkan dan mengatur ulang pembaruan untuk memodifikasi konfigurasi untuk komponen tertentu. Untuk informasi tentang `componentLogsConfiguration` parameter, lihat parameter konfigurasi untuk v2.1.x komponen ini.

`componentName`

Konfigurasi log untuk *`componentName`*komponen atau aplikasi untuk konfigurasi log ini. Anda dapat menentukan nama komponen Greengrass atau nilai lain untuk mengidentifikasi grup log ini.

Setiap objek berisi informasi berikut.

`minimumLogLevel`

(Opsional) Tingkat minimum pesan log untuk diunggah. Level minimum ini hanya berlaku jika log komponen ini menggunakan format JSON tertentu, yang dapat Anda temukan di repositori modul AWS IoT Greengrass logging. GitHub

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`diskSpaceLimit`

(Opsional) Ukuran total maksimum semua file log sistem untuk komponen ini, di unit yang Anda tentukan di `diskSpaceLimitUnit`. Setelah ukuran total file log komponen ini melebihi ukuran total maksimum ini, perangkat lunak AWS IoT Greengrass Core menghapus file log tertua komponen ini.

Parameter ini terkait dengan parameter batas ukuran log (`totalLogsSizeKB`) dari komponen inti Greengrass. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log total maksimum untuk komponen ini.

`diskSpaceLimitUnit`

(Opsional) Unit untuk `diskSpaceLimit`. Pilih dari salah satu pilihan berikut:

- `KB` – kilobyte
- `MB` – megabyte
- `GB` – gigabyte

Default: `KB`

`logFileDirectoryPath`

(Opsional) Path ke folder yang berisi file log komponen ini.

Anda tidak perlu menentukan parameter ini untuk komponen Greengrass yang mencetak pada output standar (stdout) dan kesalahan standar (stderr).

Bawaan: */greengrass/v2*`/logs`.

`logFileRegex`

(Opsional) Ekspresi reguler yang menentukan format nama file log yang menggunakan komponen atau aplikasi. Komponen manajer log menggunakan ekspresi reguler ini untuk mengidentifikasi file log dalam folder di `logFileDirectoryPath`.

Anda tidak perlu menentukan parameter ini untuk komponen Greengrass yang mencetak pada output standar (stdout) dan kesalahan standar (stderr).

Jika komponen atau aplikasi Anda memutar file log, tentukan regex yang cocok dengan nama file log yang diputar. Misalnya, Anda dapat menentukan **`hello_world\\\\w*.log`** untuk meng-upload log untuk aplikasi Hello World. Pola `\\\\w*` cocok dengan nol atau lebih karakter kata, yang meliputi karakter alfanumerik dan garis bawah. Regex ini cocok dengan file log dengan dan tanpa cap waktu dalam namanya. Dalam contoh ini, manajer log meng-upload file log berikut:

- `hello_world.log` – Berkas log terbaru untuk aplikasi Hello World.
- `hello_world_2020_12_15_17_0.log` – Berkas log yang lebih lama untuk aplikasi Hello World.

Default: *componentName*`\\\\w*.log`, tempat *componentName* adalah nama komponen untuk konfigurasi log ini.

`deleteLogFileAfterCloudUpload`

(Opsional) Anda dapat menghapus file log setelah komponen pengelola log mengunggah CloudWatch log ke Log.

Default: `false`

`multiLineStartPattern`

(Opsional) Ekspresi reguler yang mengidentifikasi kapan pesan log pada baris baru adalah pesan log baru. Jika ekspresi reguler tidak cocok dengan baris baru, komponen pengelola log menambahkan baris baru ke pesan log untuk baris sebelumnya.

Secara default, komponen pengelola log memeriksa apakah baris dimulai dengan karakter spasi putih, seperti tab atau spasi. Jika tidak, pengelola log menangani baris itu sebagai pesan log baru. Jika tidak, itu menambahkan baris itu ke pesan log saat ini. Perilaku ini memastikan bahwa komponen pengelola log tidak membagi pesan yang menjangkau beberapa baris, seperti jejak tumpukan.

`periodicUploadIntervalSec`

(Opsional) Periode dalam detik di mana komponen manajer log memeriksa file log baru yang akan diunggah.

Default: `300` (5 menit)

Minimal: `0.000001` (1 mikrodetik)

`deprecatedVersionSupport`

Menunjukkan apakah pengelola log harus menggunakan peningkatan kecepatan pencatatan yang diperkenalkan di pengelola log v2.3.5. Tetapkan nilai `false` untuk menggunakan perbaikan.

Jika Anda menetapkan nilai ini `false` saat Anda memutakhirkan dari pengelola log v2.3.1 atau entri log duplikat sebelumnya dapat diunggah.

Default-nya adalah `true`.

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk meng-upload log sistem dan log `com.example.HelloWorld` komponen ke CloudWatch Log.

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true",
      "minimumLogLevel": "INFO",
      "diskSpaceLimit": "10",
      "diskSpaceLimitUnit": "MB",
      "deleteLogFileAfterCloudUpload": "false"
    },
    "componentLogsConfigurationMap": {
      "com.example.HelloWorld": {
        "minimumLogLevel": "INFO",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
      }
    }
  },
  "periodicUploadIntervalSec": "300",
  "deprecatedVersionSupport": "false"
}
```

Example Contoh: Konfigurasi untuk mengunggah beberapa file log aktif menggunakan pengelola log v2.3.1

Contoh konfigurasi berikut adalah contoh yang disarankan jika Anda ingin menargetkan beberapa file log aktif. Konfigurasi contoh ini menentukan file log aktif apa yang ingin Anda unggah. CloudWatch Menggunakan konfigurasi contoh konfigurasi ini juga akan mengunggah file yang diputar yang cocok dengan file. `logFileRegex` Contoh konfigurasi ini didukung pada pengelola log v2.3.1.

```
{
  "logsUploaderConfiguration": {
    "componentLogsConfigurationMap": {
      "com.example.A": {
        "logFileRegex": "com.example.A\\w*.log",
        "deleteLogFileAfterCloudUpload": "false"
      }
      "com.example.B": {
        "logFileRegex": "com.example.B\\w*.log",
        "deleteLogFileAfterCloudUpload": "false"
      }
    }
  },
  "periodicUploadIntervalSec": "10"
}
```

v2.3.x

`logsUploaderConfiguration`

(Opsional) Konfigurasi untuk log yang di-upload oleh komponen manajer log. Objek ini berisi informasi berikut:

`systemLogsConfiguration`

(Opsional) Konfigurasi untuk log sistem perangkat lunak AWS IoT Greengrass Core, yang mencakup log dari inti Greengrass dan komponen plugin. Tentukan konfigurasi ini untuk mengaktifkan komponen pengelola log mengelola log sistem. Objek ini berisi informasi berikut:

`uploadToCloudWatch`

(Opsional) Anda dapat mengunggah log sistem ke CloudWatch Log.

Default: `false`

`minimumLogLevel`

(Opsional) Tingkat minimum pesan log untuk diunggah. Level minimum ini hanya berlaku jika Anda mengonfigurasi komponen inti Greengrass untuk mengeluarkan log format JSON. Untuk mengaktifkan log format JSON, `JSON` tentukan parameter format logging (`logging.format`).

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`diskSpaceLimit`

(Opsional) Ukuran total maksimum file log sistem Greengrass, di unit yang Anda tentukan. `diskSpaceLimitUnit` Setelah ukuran total file log sistem Greengrass melebihi ukuran total maksimum ini, AWS IoT Greengrass perangkat lunak Core menghapus file log sistem Greengrass tertua.

Parameter ini setara dengan parameter batas ukuran log (`totalLogsSizeKB`) dari komponen inti Greengrass. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log sistem Greengrass total maksimum.

`diskSpaceLimitUnit`

(Opsional) Unit untuk `diskSpaceLimit`. Pilih dari salah satu pilihan berikut:

- `KB` – kilobyte
- `MB` – megabyte
- `GB` – gigabyte

Default: `KB`

`deleteLogFileAfterCloudUpload`

(Opsional) Anda dapat menghapus file log setelah komponen pengelola log mengunggah CloudWatch log ke Log.

Default: `false`

`componentLogsConfigurationMap`

(Opsional) Peta konfigurasi log untuk komponen pada perangkat inti. Setiap `componentName` objek dalam peta ini mendefinisikan konfigurasi log untuk komponen atau aplikasi. Komponen pengelola log mengunggah log komponen ini ke CloudWatch Log.

> **Important**
>
> Kami sangat menyarankan menggunakan satu kunci konfigurasi per komponen. Anda hanya boleh menargetkan sekelompok file yang hanya memiliki satu file log yang secara aktif ditulis saat menggunakan file`logFileRegex`. Tidak mengikuti rekomendasi ini dapat menyebabkan log duplikat diunggah. CloudWatch Jika Anda menargetkan beberapa file log aktif dengan satu regex, kami sarankan Anda meningkatkan ke pengelola log v2.3.1 dan pertimbangkan untuk mengubah konfigurasi Anda menggunakan konfigurasi contoh.

> **Note**
>
> Jika Anda memutakhirkan dari versi pengelola log lebih awal dari v2.2.0, Anda dapat terus menggunakan `componentLogsConfiguration` daftar alih-alih. `componentLogsConfigurationMap` Namun, kami sangat menyarankan Anda menggunakan format peta sehingga Anda dapat menggunakan menggabungkan dan mengatur ulang pembaruan untuk memodifikasi konfigurasi untuk komponen tertentu. Untuk informasi tentang `componentLogsConfiguration` parameter, lihat parameter konfigurasi untuk v2.1.x komponen ini.

`componentName`

Konfigurasi log untuk *`componentName`*komponen atau aplikasi untuk konfigurasi log ini. Anda dapat menentukan nama komponen Greengrass atau nilai lain untuk mengidentifikasi grup log ini.

Setiap objek berisi informasi berikut.

`minimumLogLevel`

(Opsional) Tingkat minimum pesan log untuk diunggah. Level minimum ini hanya berlaku jika log komponen ini menggunakan format JSON tertentu, yang dapat Anda temukan di repositori modul AWS IoT Greengrass logging. GitHub

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`diskSpaceLimit`

(Opsional) Ukuran total maksimum semua file log sistem untuk komponen ini, di unit yang Anda tentukan di `diskSpaceLimitUnit`. Setelah ukuran total file log komponen ini melebihi ukuran total maksimum ini, perangkat lunak AWS IoT Greengrass Core menghapus file log tertua komponen ini.

Parameter ini terkait dengan parameter batas ukuran log (`totalLogsSizeKB`) dari komponen inti Greengrass. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log total maksimum untuk komponen ini.

`diskSpaceLimitUnit`

(Opsional) Unit untuk `diskSpaceLimit`. Pilih dari salah satu pilihan berikut:

- `KB` – kilobyte
- `MB` – megabyte
- `GB` – gigabyte

Default: `KB`

`logFileDirectoryPath`

(Opsional) Path ke folder yang berisi file log komponen ini.

Anda tidak perlu menentukan parameter ini untuk komponen Greengrass yang mencetak pada output standar (stdout) dan kesalahan standar (stderr).

Bawaan: */greengrass/v2*/logs.

`logFileRegex`

(Opsional) Ekspresi reguler yang menentukan format nama file log yang menggunakan komponen atau aplikasi. Komponen manajer log menggunakan ekspresi reguler ini untuk mengidentifikasi file log dalam folder di `logFileDirectoryPath`.

Anda tidak perlu menentukan parameter ini untuk komponen Greengrass yang mencetak pada output standar (stdout) dan kesalahan standar (stderr).

Jika komponen atau aplikasi Anda memutar file log, tentukan regex yang cocok dengan nama file log yang diputar. Misalnya, Anda dapat menentukan **`hello_world\\\\w*.log`** untuk meng-upload log untuk aplikasi Hello World. Pola `\\\\w*` cocok dengan nol atau lebih karakter kata, yang meliputi karakter alfanumerik dan garis bawah. Regex ini cocok dengan file log dengan dan tanpa cap waktu dalam namanya. Dalam contoh ini, manajer log meng-upload file log berikut:

- `hello_world.log` – Berkas log terbaru untuk aplikasi Hello World.
- `hello_world_2020_12_15_17_0.log` – Berkas log yang lebih lama untuk aplikasi Hello World.

Default: *componentName*`\\\\w*.log`, tempat *componentName* adalah nama komponen untuk konfigurasi log ini.

`deleteLogFileAfterCloudUpload`

(Opsional) Anda dapat menghapus file log setelah komponen pengelola log mengunggah CloudWatch log ke Log.

Default: `false`

`multiLineStartPattern`

(Opsional) Ekspresi reguler yang mengidentifikasi kapan pesan log pada baris baru adalah pesan log baru. Jika ekspresi reguler tidak cocok dengan baris baru, komponen pengelola log menambahkan baris baru ke pesan log untuk baris sebelumnya.

Secara default, komponen pengelola log memeriksa apakah baris dimulai dengan karakter spasi putih, seperti tab atau spasi. Jika tidak, pengelola log menangani baris itu sebagai pesan log baru. Jika tidak, itu menambahkan baris itu ke pesan

log saat ini. Perilaku ini memastikan bahwa komponen pengelola log tidak membagi pesan yang menjangkau beberapa baris, seperti jejak tumpukan.

`periodicUploadIntervalSec`

(Opsional) Periode dalam detik di mana komponen manajer log memeriksa file log baru yang akan diunggah.

Default: `300` (5 menit)

Minimal: `0.000001` (1 mikrodetik)

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk meng-upload log sistem dan log `com.example.HelloWorld` komponen ke CloudWatch Log.

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true",
      "minimumLogLevel": "INFO",
      "diskSpaceLimit": "10",
      "diskSpaceLimitUnit": "MB",
      "deleteLogFileAfterCloudUpload": "false"
    },
    "componentLogsConfigurationMap": {
      "com.example.HelloWorld": {
        "minimumLogLevel": "INFO",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
      }
    }
  },
  "periodicUploadIntervalSec": "300"
}
```

Example Contoh: Konfigurasi untuk mengunggah beberapa file log aktif menggunakan pengelola log v2.3.1

Contoh konfigurasi berikut adalah contoh yang disarankan jika Anda ingin menargetkan beberapa file log aktif. Konfigurasi contoh ini menentukan file log aktif apa yang ingin Anda unggah.

CloudWatch Menggunakan konfigurasi contoh konfigurasi ini juga akan mengunggah file yang diputar yang cocok dengan file. `logFileRegex` Contoh konfigurasi ini didukung pada pengelola log v2.3.1.

```
{
  "logsUploaderConfiguration": {
    "componentLogsConfigurationMap": {
      "com.example.A": {
        "logFileRegex": "com.example.A\\w*.log",
        "deleteLogFileAfterCloudUpload": "false"
      }
      "com.example.B": {
        "logFileRegex": "com.example.B\\w*.log",
        "deleteLogFileAfterCloudUpload": "false"
      }
    }
  },
  "periodicUploadIntervalSec": "10"
}
```

v2.2.x

`logsUploaderConfiguration`

(Opsional) Konfigurasi untuk log yang di-upload oleh komponen manajer log. Objek ini berisi informasi berikut:

`systemLogsConfiguration`

(Opsional) Konfigurasi untuk log sistem perangkat lunak AWS IoT Greengrass Core, yang mencakup log dari inti Greengrass dan komponen plugin. Tentukan konfigurasi ini untuk mengaktifkan komponen pengelola log mengelola log sistem. Objek ini berisi informasi berikut:

`uploadToCloudWatch`

(Opsional) Anda dapat mengunggah log sistem ke CloudWatch Log.

Default: `false`

`minimumLogLevel`

(Opsional) Tingkat minimum pesan log untuk diunggah. Level minimum ini hanya berlaku jika Anda mengonfigurasi komponen inti Greengrass untuk mengeluarkan log

format JSON. Untuk mengaktifkan log format JSON, `JSON` tentukan parameter format logging (`logging.format`).

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`diskSpaceLimit`

(Opsional) Ukuran total maksimum file log sistem Greengrass, di unit yang Anda tentukan. `diskSpaceLimitUnit` Setelah ukuran total file log sistem Greengrass melebihi ukuran total maksimum ini, AWS IoT Greengrass perangkat lunak Core menghapus file log sistem Greengrass tertua.

Parameter ini setara dengan parameter batas ukuran log (`totalLogsSizeKB`) dari komponen inti Greengrass. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log sistem Greengrass total maksimum.

`diskSpaceLimitUnit`

(Opsional) Unit untuk `diskSpaceLimit`. Pilih dari salah satu pilihan berikut:

- `KB` – kilobyte
- `MB` – megabyte
- `GB` – gigabyte

Default: `KB`

`deleteLogFileAfterCloudUpload`

(Opsional) Anda dapat menghapus file log setelah komponen pengelola log mengunggah CloudWatch log ke Log.

Default: `false`

`componentLogsConfigurationMap`

(Opsional) Peta konfigurasi log untuk komponen pada perangkat inti. Setiap `componentName` objek dalam peta ini mendefinisikan konfigurasi log untuk komponen atau aplikasi. Komponen pengelola log mengunggah log komponen ini ke CloudWatch Log.

> **Note**
>
> Jika Anda memutakhirkan dari versi pengelola log lebih awal dari v2.2.0, Anda dapat terus menggunakan `componentLogsConfiguration` daftar alih-alih. `componentLogsConfigurationMap` Namun, kami sangat menyarankan Anda menggunakan format peta sehingga Anda dapat menggunakan menggabungkan dan mengatur ulang pembaruan untuk memodifikasi konfigurasi untuk komponen tertentu. Untuk informasi tentang `componentLogsConfiguration` parameter, lihat parameter konfigurasi untuk v2.1.x komponen ini.

`componentName`

Konfigurasi log untuk *`componentName`*komponen atau aplikasi untuk konfigurasi log ini. Anda dapat menentukan nama komponen Greengrass atau nilai lain untuk mengidentifikasi grup log ini.

Setiap objek berisi informasi berikut.

`minimumLogLevel`

(Opsional) Tingkat minimum pesan log untuk diunggah. Level minimum ini hanya berlaku jika log komponen ini menggunakan format JSON tertentu, yang dapat Anda temukan di repositori modul AWS IoT Greengrass logging. GitHub

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`diskSpaceLimit`

(Opsional) Ukuran total maksimum semua file log sistem untuk komponen ini, di unit yang Anda tentukan di `diskSpaceLimitUnit`. Setelah ukuran total file log komponen ini melebihi ukuran total maksimum ini, perangkat lunak AWS IoT Greengrass Core menghapus file log tertua komponen ini.

Parameter ini terkait dengan parameter batas ukuran log (`totalLogsSizeKB`) dari komponen inti Greengrass. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log total maksimum untuk komponen ini.

`diskSpaceLimitUnit`

(Opsional) Unit untuk `diskSpaceLimit`. Pilih dari salah satu pilihan berikut:

- `KB` – kilobyte
- `MB` – megabyte
- `GB` – gigabyte

Default: `KB`

`logFileDirectoryPath`

(Opsional) Path ke folder yang berisi file log komponen ini.

Anda tidak perlu menentukan parameter ini untuk komponen Greengrass yang mencetak pada output standar (stdout) dan kesalahan standar (stderr).

Bawaan: */greengrass/v2*`/logs`.

`logFileRegex`

(Opsional) Ekspresi reguler yang menentukan format nama file log yang menggunakan komponen atau aplikasi. Komponen manajer log menggunakan ekspresi reguler ini untuk mengidentifikasi file log dalam folder di `logFileDirectoryPath`.

Anda tidak perlu menentukan parameter ini untuk komponen Greengrass yang mencetak pada output standar (stdout) dan kesalahan standar (stderr).

Jika komponen atau aplikasi Anda memutar file log, tentukan regex yang cocok dengan nama file log yang diputar. Misalnya, Anda dapat menentukan

`hello_world\\\\w*.log` untuk meng-upload log untuk aplikasi Hello World. Pola `\\\\w*` cocok dengan nol atau lebih karakter kata, yang meliputi karakter alfanumerik dan garis bawah. Regex ini cocok dengan file log dengan dan tanpa cap waktu dalam namanya. Dalam contoh ini, manajer log meng-upload file log berikut:

- `hello_world.log` – Berkas log terbaru untuk aplikasi Hello World.
- `hello_world_2020_12_15_17_0.log` – Berkas log yang lebih lama untuk aplikasi Hello World.

Default: *`componentName`*`\\\\w*.log`, tempat *`componentName`* adalah nama komponen untuk konfigurasi log ini.

`deleteLogFileAfterCloudUpload`

(Opsional) Anda dapat menghapus file log setelah komponen pengelola log mengunggah CloudWatch log ke Log.

Default: `false`

`multiLineStartPattern`

(Opsional) Ekspresi reguler yang mengidentifikasi kapan pesan log pada baris baru adalah pesan log baru. Jika ekspresi reguler tidak cocok dengan baris baru, komponen pengelola log menambahkan baris baru ke pesan log untuk baris sebelumnya.

Secara default, komponen pengelola log memeriksa apakah baris dimulai dengan karakter spasi putih, seperti tab atau spasi. Jika tidak, pengelola log menangani baris itu sebagai pesan log baru. Jika tidak, itu menambahkan baris itu ke pesan log saat ini. Perilaku ini memastikan bahwa komponen pengelola log tidak membagi pesan yang menjangkau beberapa baris, seperti jejak tumpukan.

`periodicUploadIntervalSec`

(Opsional) Periode dalam detik di mana komponen manajer log memeriksa file log baru yang akan diunggah.

Default: `300` (5 menit)

Minimal: `0.000001` (1 mikrodetik)

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk meng-upload log sistem dan log `com.example.HelloWorld` komponen ke CloudWatch Log.

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true",
      "minimumLogLevel": "INFO",
      "diskSpaceLimit": "10",
      "diskSpaceLimitUnit": "MB",
      "deleteLogFileAfterCloudUpload": "false"
    },
    "componentLogsConfigurationMap": {
      "com.example.HelloWorld": {
        "minimumLogLevel": "INFO",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
      }
    }
  },
  "periodicUploadIntervalSec": "300"
}
```

v2.1.x

`logsUploaderConfiguration`

(Opsional) Konfigurasi untuk log yang di-upload oleh komponen manajer log. Objek ini berisi informasi berikut:

`systemLogsConfiguration`

(Opsional) Konfigurasi untuk log sistem perangkat lunak AWS IoT Greengrass Core, yang mencakup log dari inti Greengrass dan komponen plugin. Tentukan konfigurasi ini untuk mengaktifkan komponen pengelola log mengelola log sistem. Objek ini berisi informasi berikut:

`uploadToCloudWatch`

(Opsional) Anda dapat mengunggah log sistem ke CloudWatch Log.

Default: `false`

`minimumLogLevel`

(Opsional) Tingkat minimum pesan log untuk diunggah. Level minimum ini hanya berlaku jika Anda mengonfigurasi komponen inti Greengrass untuk mengeluarkan log format JSON. Untuk mengaktifkan log format JSON, `JSON` tentukan parameter format logging (`logging.format`).

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`diskSpaceLimit`

(Opsional) Ukuran total maksimum file log sistem Greengrass, di unit yang Anda tentukan. `diskSpaceLimitUnit` Setelah ukuran total file log sistem Greengrass melebihi ukuran total maksimum ini, AWS IoT Greengrass perangkat lunak Core menghapus file log sistem Greengrass tertua.

Parameter ini setara dengan parameter batas ukuran log (`totalLogsSizeKB`) dari komponen inti Greengrass. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log sistem Greengrass total maksimum.

`diskSpaceLimitUnit`

(Opsional) Unit untuk `diskSpaceLimit`. Pilih dari salah satu pilihan berikut:

- `KB` – kilobyte
- `MB` – megabyte
- `GB` – gigabyte

Default: `KB`

`deleteLogFileAfterCloudUpload`

(Opsional) Anda dapat menghapus file log setelah komponen pengelola log mengunggah CloudWatch log ke Log.

Default: `false`

`componentLogsConfiguration`

(Opsional) Daftar konfigurasi log untuk komponen pada perangkat inti. Setiap konfigurasi dalam daftar ini mendefinisikan konfigurasi log untuk komponen atau aplikasi. Komponen pengelola log mengunggah log komponen ini ke CloudWatch Log

Setiap objek berisi informasi berikut.

`componentName`

Nama komponen atau aplikasi untuk konfigurasi log ini. Anda dapat menentukan nama komponen Greengrass atau nilai lain untuk mengidentifikasi grup log ini.

`minimumLogLevel`

(Opsional) Tingkat minimum pesan log untuk diunggah. Level minimum ini hanya berlaku jika log komponen ini menggunakan format JSON tertentu, yang dapat Anda temukan di repositori modul AWS IoT Greengrass logging. GitHub

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`diskSpaceLimit`

(Opsional) Ukuran total maksimum semua file log sistem untuk komponen ini, di unit yang Anda tentukan di `diskSpaceLimitUnit`. Setelah ukuran total file log komponen ini melebihi ukuran total maksimum ini, perangkat lunak AWS IoT Greengrass Core menghapus file log tertua komponen ini.

Parameter ini terkait dengan parameter batas ukuran log (`totalLogsSizeKB`) dari komponen inti Greengrass. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log total maksimum untuk komponen ini.

`diskSpaceLimitUnit`

(Opsional) Unit untuk `diskSpaceLimit`. Pilih dari salah satu pilihan berikut:

- `KB` – kilobyte
- `MB` – megabyte
- `GB` – gigabyte

Default: `KB`

`logFileDirectoryPath`

(Opsional) Path ke folder yang berisi file log komponen ini.

Anda tidak perlu menentukan parameter ini untuk komponen Greengrass yang mencetak pada output standar (stdout) dan kesalahan standar (stderr).

Bawaan: *`/greengrass/v2`*`/logs`.

`logFileRegex`

(Opsional) Ekspresi reguler yang menentukan format nama file log yang menggunakan komponen atau aplikasi. Komponen manajer log menggunakan ekspresi reguler ini untuk mengidentifikasi file log dalam folder di `logFileDirectoryPath`.

Anda tidak perlu menentukan parameter ini untuk komponen Greengrass yang mencetak pada output standar (stdout) dan kesalahan standar (stderr).

Jika komponen atau aplikasi Anda memutar file log, tentukan regex yang cocok dengan nama file log yang diputar. Misalnya, Anda dapat menentukan **`hello_world\\\\w*.log`** untuk meng-upload log untuk aplikasi Hello World. Pola `\\\\w*` cocok dengan nol atau lebih karakter kata, yang meliputi karakter alfanumerik dan garis bawah. Regex ini cocok dengan file log dengan dan tanpa cap waktu dalam namanya. Dalam contoh ini, manajer log meng-upload file log berikut:

- `hello_world.log` – Berkas log terbaru untuk aplikasi Hello World.
- `hello_world_2020_12_15_17_0.log` – Berkas log yang lebih lama untuk aplikasi Hello World.

Default: *`componentName`*`\\\\w*.log`, tempat *`componentName`* adalah nama komponen untuk konfigurasi log ini.

`deleteLogFileAfterCloudUpload`

(Opsional) Anda dapat menghapus file log setelah komponen pengelola log mengunggah CloudWatch log ke Log.

Default: `false`

`multiLineStartPattern`

(Opsional) Ekspresi reguler yang mengidentifikasi kapan pesan log pada baris baru adalah pesan log baru. Jika ekspresi reguler tidak cocok dengan baris baru, komponen pengelola log menambahkan baris baru ke pesan log untuk baris sebelumnya.

Secara default, komponen pengelola log memeriksa apakah baris dimulai dengan karakter spasi putih, seperti tab atau spasi. Jika tidak, pengelola log menangani baris itu sebagai pesan log baru. Jika tidak, itu menambahkan baris itu ke pesan log saat ini. Perilaku ini memastikan bahwa komponen pengelola log tidak membagi pesan yang menjangkau beberapa baris, seperti jejak tumpukan.

`periodicUploadIntervalSec`

(Opsional) Periode dalam detik di mana komponen manajer log memeriksa file log baru yang akan diunggah.

Default: `300` (5 menit)

Minimal: `0.000001` (1 mikrodetik)

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk meng-upload log sistem dan log `com.example.HelloWorld` komponen ke CloudWatch Log.

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true",
      "minimumLogLevel": "INFO",
      "diskSpaceLimit": "10",
      "diskSpaceLimitUnit": "MB",
      "deleteLogFileAfterCloudUpload": "false"
    },
    "componentLogsConfiguration": [
      {
        "componentName": "com.example.HelloWorld",
        "minimumLogLevel": "INFO",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
```

```
      }
    ]
  },
  "periodicUploadIntervalSec": "300"
}
```

v2.0.x

`logsUploaderConfiguration`

(Opsional) Konfigurasi untuk log yang di-upload oleh komponen manajer log. Objek ini berisi informasi berikut:

`systemLogsConfiguration`

(Opsional) Konfigurasi untuk log sistem perangkat lunak inti AWS IoT Greengrass. Tentukan konfigurasi ini untuk mengaktifkan komponen pengelola log mengelola log sistem. Objek ini berisi informasi berikut:

`uploadToCloudWatch`

(Opsional) Anda dapat mengunggah log sistem ke CloudWatch Log.

Default: `false`

`minimumLogLevel`

(Opsional) Tingkat minimum pesan log untuk diunggah. Level minimum ini hanya berlaku jika Anda mengonfigurasi komponen inti Greengrass untuk mengeluarkan log format JSON. Untuk mengaktifkan log format JSON, `JSON` tentukan parameter format logging (`logging.format`).

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`diskSpaceLimit`

(Opsional) Ukuran total maksimum file log sistem Greengrass, di unit yang Anda tentukan. `diskSpaceLimitUnit` Setelah ukuran total file log sistem Greengrass

melebihi ukuran total maksimum ini, AWS IoT Greengrass perangkat lunak Core menghapus file log sistem Greengrass tertua.

Parameter ini setara dengan parameter batas ukuran log (`totalLogsSizeKB`) dari komponen inti Greengrass. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log sistem Greengrass total maksimum.

`diskSpaceLimitUnit`

(Opsional) Unit untuk `diskSpaceLimit`. Pilih dari salah satu pilihan berikut:

- `KB` – kilobyte
- `MB` – megabyte
- `GB` – gigabyte

Default: `KB`

`deleteLogFileAfterCloudUpload`

(Opsional) Anda dapat menghapus file log setelah komponen pengelola log mengunggah CloudWatch log ke Log.

Default: `false`

`componentLogsConfiguration`

(Opsional) Daftar konfigurasi log untuk komponen pada perangkat inti. Setiap konfigurasi dalam daftar ini mendefinisikan konfigurasi log untuk komponen atau aplikasi. Komponen pengelola log mengunggah log komponen ini ke CloudWatch Log

Setiap objek berisi informasi berikut.

`componentName`

Nama komponen atau aplikasi untuk konfigurasi log ini. Anda dapat menentukan nama komponen Greengrass atau nilai lain untuk mengidentifikasi grup log ini.

`minimumLogLevel`

(Opsional) Tingkat minimum pesan log untuk diunggah. Level minimum ini hanya berlaku jika log komponen ini menggunakan format JSON tertentu, yang dapat Anda temukan di repositori modul AWS IoT Greengrass logging. GitHub

Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `DEBUG`

- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`diskSpaceLimit`

(Opsional) Ukuran total maksimum semua file log sistem untuk komponen ini, di unit yang Anda tentukan di `diskSpaceLimitUnit`. Setelah ukuran total file log komponen ini melebihi ukuran total maksimum ini, perangkat lunak AWS IoT Greengrass Core menghapus file log tertua komponen ini.

Parameter ini terkait dengan parameter batas ukuran log (`totalLogsSizeKB`) dari komponen inti Greengrass. Perangkat lunak AWS IoT Greengrass Core menggunakan minimum dua nilai sebagai ukuran log total maksimum untuk komponen ini.

`diskSpaceLimitUnit`

(Opsional) Unit untuk `diskSpaceLimit`. Pilih dari salah satu pilihan berikut:

- `KB` – kilobyte
- `MB` – megabyte
- `GB` – gigabyte

Default: `KB`

`logFileDirectoryPath`

Path ke folder yang berisi file log komponen ini.

Untuk mengunggah log komponen Greengrass, ***/greengrass/v2*/logs** tentukan, dan ganti */greengrass/v2* dengan folder root Greengrass Anda.

`logFileRegex`

Ekspresi reguler yang menentukan format nama file log yang digunakan oleh komponen atau aplikasi. Komponen manajer log menggunakan ekspresi reguler ini untuk mengidentifikasi file log dalam folder di `logFileDirectoryPath`.

Untuk meng-upload log komponen Greengrass tentukan regex yang cocok dengan nama file log yang diputar. Misalnya, Anda dapat menentukan **`com.example.HelloWorld\\w*.log`** untuk meng-upload log untuk aplikasi Hello World. Pola `\\w*` cocok dengan nol atau lebih karakter kata, yang meliputi karakter

alfanumerik dan garis bawah. Regex ini cocok dengan file log dengan dan tanpa cap waktu dalam namanya. Dalam contoh ini, manajer log meng-upload file log berikut:

- `com.example.HelloWorld.log` – Berkas log terbaru untuk komponen Hello World.
- `com.example.HelloWorld_2020_12_15_17_0.log` – Berkas log yang lebih lama untuk komponen Hello World. Inti Greengrass menambahkan stempel waktu putar pada file log.

`deleteLogFileAfterCloudUpload`

(Opsional) Anda dapat menghapus file log setelah komponen pengelola log mengunggah CloudWatch log ke Log.

Default: `false`

`multiLineStartPattern`

(Opsional) Ekspresi reguler yang mengidentifikasi kapan pesan log pada baris baru adalah pesan log baru. Jika ekspresi reguler tidak cocok dengan baris baru, komponen pengelola log menambahkan baris baru ke pesan log untuk baris sebelumnya.

Secara default, komponen pengelola log memeriksa apakah baris dimulai dengan karakter spasi putih, seperti tab atau spasi. Jika tidak, pengelola log menangani baris itu sebagai pesan log baru. Jika tidak, itu menambahkan baris itu ke pesan log saat ini. Perilaku ini memastikan bahwa komponen pengelola log tidak membagi pesan yang menjangkau beberapa baris, seperti jejak tumpukan.

`periodicUploadIntervalSec`

(Opsional) Periode dalam detik di mana komponen manajer log memeriksa file log baru yang akan diunggah.

Default: `300` (5 menit)

Minimal: `0.000001` (1 mikrodetik)

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk meng-upload log sistem dan log `com.example.HelloWorld` komponen ke CloudWatch Log.

```
{
```

```
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true",
      "minimumLogLevel": "INFO",
      "diskSpaceLimit": "10",
      "diskSpaceLimitUnit": "MB",
      "deleteLogFileAfterCloudUpload": "false"
    },
    "componentLogsConfiguration": [
      {
        "componentName": "com.example.HelloWorld",
        "minimumLogLevel": "INFO",
        "logFileDirectoryPath": "/greengrass/v2/logs",
        "logFileRegex": "com.example.HelloWorld\\w*.log",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
      }
    ]
  },
  "periodicUploadIntervalSec": "300"
}
```

Penggunaan

Komponen manajer log meng-upload ke grup log dan pengaliran log berikut.

2.1.0 and later

Nama grup log

```
/aws/greengrass/componentType/region/componentName
```

Nama grup log menggunakan variabel berikut:

- `componentType` — Jenis komponen, yang dapat berupa salah satu dari berikut ini:
  - `GreengrassSystemComponent`— Grup log ini mencakup log untuk komponen inti dan plugin, yang berjalan di JVM yang sama dengan inti Greengrass. Komponen ini merupakan bagian dari inti Greengrass.
  - `UserComponent`— Grup log ini mencakup log untuk komponen generik, komponen Lambda, dan aplikasi lain di perangkat. Komponen ini bukan bagian dari inti Greengrass.

Untuk informasi selengkapnya, lihat Jenis komponen.

- `region` — Wilayah AWS yang menggunakan perangkat inti.
- `componentName` — Nama komponen. Untuk log sistem, nilai ini adalah `System`.

Nama aliran log

```
/date/thing/thingName
```

Nama aliran log menggunakan variabel berikut:

- `date` — Tanggal log, seperti `2020/12/15`. Komponen manajer log menggunakan format `yyyy/MM/dd`.
- `thingName` - Nama perangkat inti.

> **Note**
>
> Jika nama objek berisi titik dua (`:`), manajer log akan menggantikan titik dua dengan tanda tambah (+).

2.0.x

Nama grup log

```
/aws/greengrass/componentType/region/componentName
```

Nama grup log menggunakan variabel berikut:

- `componentType` — Jenis komponen, yang dapat berupa salah satu dari berikut ini:
  - `GreengrassSystemComponent` — Komponen adalah bagian dari nukleus Greengrass.
  - `UserComponent` — Komponen adalah bagian dari nukleus Greengrass. Manajer log menggunakan jenis ini untuk komponen Greengrass dan aplikasi lainnya pada perangkat.
- `region` — Wilayah AWS yang menggunakan perangkat inti.
- `componentName` — Nama komponen. Untuk log sistem, nilai ini adalah `System`.

Nama aliran log

```
/date/deploymentTargets/thingName
```

Nama aliran log menggunakan variabel berikut:

- `date` — Tanggal log, seperti `2020/12/15`. Komponen manajer log menggunakan format `yyyy/MM/dd`.
- `deploymentTargets` — Objek-objek yang deployment-nya meliputi komponen. Komponen manajer log memisahkan setiap target dengan garis miring. Jika komponen berjalan pada perangkat inti sebagai hasil dari deployment lokal, nilai ini adalah `LOCAL_DEPLOYMENT`.

  Pertimbangkan contoh di mana Anda memiliki perangkat inti bernama `MyGreengrassCore`, dan perangkat inti tersebut memiliki dua deployment:

  - Deployment yang menargetkan perangkat inti, `MyGreengrassCore`.
  - Deployment yang menargetkan grup objek bernama `MyGreengrassCoreGroup`, yang berisi perangkat inti.

  `deploymentTargets` untuk perangkat inti ini adalah `thing/MyGreengrassCore/thinggroup/MyGreengrassCoreGroup`.
- `thingName` - Nama perangkat inti.

Format untuk entri log.

Inti Greengrass menulis file log dalam format string atau JSON. Untuk log sistem, Anda mengontrol format dengan mengatur `format` bidang `logging` entri. Anda dapat menemukan `logging` entri di file konfigurasi komponen inti Greengrass. Untuk informasi selengkapnya, lihat konfigurasi nukleus Greengrass.

Format teks adalah bentuk bebas dan menerima string apa pun. Pesan layanan status armada berikut adalah contoh logging berformat string:

```
2023-03-26T18:18:27.271Z [INFO] (pool-1-thread-2)
com.aws.greengrass.status.FleetStatusService: fss-status-update-published.
Status update published to FSS. {trigger=CADENCE, serviceName=FleetStatusService,
currentState=RUNNING}
```

Anda harus menggunakan format JSON jika Anda ingin melihat log dengan perintah log CLI Greengrass atau berinteraksi dengan log secara terprogram. Contoh berikut menguraikan bentuk JSON:

```
{
```

```
    "loggerName": <string>,
    "level": <"DEBUG" | "INFO" | "ERROR" | "TRACE" | "WARN">,
    "eventType": <string, optional>,
    "cause": <string, optional>,
    "contexts": {},
    "thread": <string>,
    "message": <string>,
    "timestamp": <epoch time> # Needs to be epoch time
}
```

Untuk mengontrol output log komponen Anda, Anda dapat menggunakan opsi `minimumLogLevel` konfigurasi. Untuk menggunakan opsi ini, komponen Anda harus menulis entri lognya dalam format JSON. Anda harus menggunakan format yang sama dengan file log sistem.

Berkas log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.3.7	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.3.6	Perbaikan bug dan peningkatan • Menyesuaikan tingkat log untuk kesalahan tertentu.
2.3.5	Perbaikan Meningkatkan kecepatan unggah log. Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.3.4	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk mengatur `periodicUploadIntervalSec` parameter ke nilai fraksional. Minimal adalah 1 mikrodetik. • Memperbaiki masalah di mana pengelola log tidak menghormati CloudWatch `putLogEvents` batas.
2.3.3	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.3.2	Perbaikan bug dan peningkatan • Meningkatkan manajemen ruang sehingga file log tidak dihapus sebelum diunggah. • Memperbaiki masalah dengan manajemen cache. • Peningkatan dan perbaikan kecil tambahan.
2.3.1	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana file target tersebut mengelompokkan dengan kelipatan file log aktif mengunggah entri duplikat ke. CloudWatch • Peningkatan dan perbaikan kecil tambahan.

Versi	Perubahan
2.3.0	**Note** Kami menyarankan Anda meningkatkan ke Greengrass nucleus 2.9.1 saat Anda meningkatkan ke pengelola log 2.3.0. Fitur baru Mengurangi penundaan log dengan memproses dan langsung mengungga h file log aktif alih-alih menunggu file baru diputar. Perbaikan bug dan peningkatan • Meningkatkan dukungan rotasi log saat memutar file dengan nama unik. • Peningkatan dan perbaikan kecil tambahan.
2.2.8	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.2.7	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.2.6	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.2.5	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.2.4	Perbaikan bug dan peningkatan • Meningkatkan stabilitas saat menangani konfigurasi yang tidak valid. • Peningkatan dan perbaikan kecil tambahan.

Versi	Perubahan
2.2.3	Perbaikan bug dan peningkatan • Meningkatkan stabilitas dalam skenario tertentu di mana komponen memulai ulang atau menemukan kesalahan. • Memperbaiki masalah di mana pesan log besar dan file log besar gagal diunggah dalam skenario tertentu. • Memperbaiki masalah dengan cara komponen ini menangani pembaruan pengaturan ulang konfigurasi. • Memperbaiki masalah di mana nilai `null` `diskSpaceLimit` konfigurasi mencegah komponen dari penerapan.
2.2.2	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk pesan log yang lebih besar dari 256 kilobyte. Komponen pengelola log membagi pesan log besar ini menjadi beberapa pesan dengan stempel waktu peristiwa log yang sama.
2.2.1	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.2.0	Fitur baru • Menambahkan parameter `componentLogsConfigurationMap` konfigurasi untuk mendukung format peta untuk konfigurasi log komponen. Setiap `componentName` objek dalam peta mendefinisikan konfigurasi log untuk komponen atau aplikasi.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.1.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.1	Perbaikan bug dan peningkatan • Memperbaiki masalah saat konfigurasi log sistem tidak diperbarui dalam kasus tertentu.

Versi	Perubahan
2.1.0	Perbaikan bug dan peningkatan • Gunakan default untuk `logFileDirectoryPath` dan `logFileRegex` yang bekerja untuk komponen Greengrass yang mencetak ke output standar (stdout) dan kesalahan standar (stderr). • Rutekan lalu lintas dengan benar melalui proxy jaringan yang dikonfigurasi saat mengunggah CloudWatch log ke Log. • Tangani karakter titik dua (`:`) dengan benar di nama aliran log. CloudWatch Nama aliran log log tidak mendukung titik dua. • Sederhanakan nama aliran log dengan menghapus nama grup objek dari aliran log. • Hapus pesan log kesalahan yang dicetak selama perilaku normal.
2.0.x	Versi awal.

Komponen machine learning

AWS IoT Greengrassmenyediakan komponen pembelajaran mesin berikut yang dapat Anda terapkan ke perangkat yang didukung untuk melakukan inferensi pembelajaran mesin menggunakan model yang dilatih di Amazon SageMaker atau dengan model terlatih Anda sendiri yang disimpan di Amazon S3.

AWSmenyediakan kategori komponen pembelajaran mesin berikut:

- Komponen model—Berisi model machine learning sebagai artefak Greengrass.
- Komponen waktu aktif—Berisi skrip yang menginstal kerangka kerja machine learning dan dependensinya pada perangkat inti Greengrass.
- Komponen Inferensi—Berisi kode inferensi dan mencakup dependensi komponen untuk menginstal kerangka machine learning dan mengunduh model machine learning yang telah dilatih sebelumnya.

Anda dapat menggunakan kode inferensi sampel dan model yang telah dilatih sebelumnya dalam komponen pembelajaran mesin AWS yang disediakan untuk melakukan klasifikasi gambar dan deteksi objek menggunakan DLR dan Lite. TensorFlow Untuk melakukan inferensi pembelajaran

mesin khusus dengan model Anda sendiri yang disimpan di Amazon S3, atau menggunakan kerangka kerja pembelajaran mesin yang berbeda, Anda dapat menggunakan resep komponen publik ini sebagai templat untuk membuat komponen pembelajaran mesin khusus. Untuk informasi selengkapnya, lihat Sesuaikan komponen machine learning Anda.

AWS IoT Greengrassjuga mencakup komponen AWS yang disediakan untuk mengelola instalasi dan siklus hidup agen SageMaker Edge Manager pada perangkat inti Greengrass. Dengan SageMaker Edge Manager, Anda dapat menggunakan model Amazon SageMaker Neo yang dikompilasi langsung di perangkat inti Anda. Untuk informasi selengkapnya, lihat Gunakan Amazon SageMaker Edge Manager di perangkat inti Greengrass.

Tabel berikut mencantumkan komponen pembelajaran mesin yang tersedia diAWS IoT Greengrass.

> **Note**
>
> Beberapa komponen AWS yang disediakan bergantung pada versi minor spesifik dari inti Greengrass. Karena ketergantungan ini, Anda perlu memperbarui komponen ini saat memperbarui inti Greengrass ke versi minor baru. Untuk informasi tentang versi spesifik dari inti yang masing-masing komponen bergantung padanya, lihat topik komponen yang sesuai. Untuk informasi selengkapnya terkait cara memperbarui inti, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Ketika komponen memiliki tipe komponen generik dan Lambda, versi komponen saat ini adalah tipe generik dan versi komponen sebelumnya adalah tipe Lambda.

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Lookout for Vision Edge Agent	Menyebarkan runtime Amazon Lookout for Vision pada perangkat inti Greengras	Generik	Linux	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
	s, sehingga Anda dapat menggunakan visi komputer untuk menemukan cacat pada produk industri.			
SageMaker Manajer Tepi	Menyebarkan agen Amazon SageMaker Edge Manager di perangkat inti Greengras s.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Klasifikasi citra DLR	Komponen inferensi yang menggunakan penyimpanan model klasifikasi gambar DLR dan komponen waktu aktif DLR sebagai dependensi akan menginstal DLR, men-download model klasifikasi gambar sampel, dan melakukan inferensi klasifikasi gambar pada perangkat yang didukung.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Deteksi objek DLR	Komponen inferensi yang menggunakan penyimpanan model deteksi gambar DLR dan komponen waktu aktif DLR sebagai dependens i akan menginstal DLR, men-download sampel model deteksi, dan melakukan inferensi deteksi gambar pada perangkat yang didukung.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Penyimpanan model klasifikasi gambar DLR	Komponen model yang berisi sampel ResNet -50 model klasifikasi gambar sebagai artefak Greengrass.	Generik	Linux, Windows	Tidak
Penyimpanan model deteksi DLR	Komponen model yang berisi sampel model deteksi objek YOLOv3 sebagai artefak Greengrass.	Generik	Linux, Windows	Tidak
Runtime DLR	Komponen waktu aktif yang berisi skrip instalasi yang digunakan untuk menginstal DLR dan dependensinya pada perangkat inti Greengrass.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
TensorFlow Klasifikasi gambar ringan	Komponen inferensi yang menggunakan penyimpanan model klasifikasi gambar TensorFlow Lite dan komponen runtime TensorFlow Lite sebagai dependensi untuk menginstal TensorFlow Lite, mengunduh model klasifikasi gambar sampel, dan melakukan inferensi klasifikasi gambar pada perangkat yang didukung.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
TensorFlow Deteksi objek Lite	Komponen inferensi yang menggunakan penyimpanan model deteksi objek TensorFlow Lite dan komponen runtime TensorFlow Lite sebagai dependensi untuk menginstal TensorFlow Lite, mengunduh model deteksi objek sampel, dan melakukan inferensi deteksi objek pada perangkat yang didukung.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
TensorFlow Toko model klasifikasi gambar Lite	Komponen model yang berisi contoh model MobileNet v1 sebagai artefak Greengrass.	Generik	Linux, Windows	Tidak
TensorFlow Toko model deteksi objek Lite	Komponen model yang berisi contoh model Single Shot Detection (SSD) sebagai MobileNet artefak Greengrass.	Generik	Linux, Windows	Tidak
TensorFlow Runtime ringan	Komponen runtime yang berisi skrip instalasi yang digunakan untuk menginstal TensorFlow Lite dan dependensinya pada perangkat inti Greengrass.	Generik	Linux, Windows	Tidak

Lookout for Vision Edge Agent

Komponen Lookout for Vision Edge Agent `aws.iot.lookoutvision.EdgeAgent` () menginstal server runtime Amazon Lookout for Vision lokal, yang menggunakan visi komputer untuk menemukan cacat visual pada produk industri.

Untuk menggunakan komponen ini, buat dan terapkan komponen model machine learning Lookout for Vision. Model pembelajaran mesin ini memprediksi adanya anomali dalam gambar dengan menemukan pola dalam gambar yang Anda gunakan untuk melatih model. Kemudian, Anda dapat mengembangkan dan menerapkan komponen Greengrass kustom, yang disebut komponen aplikasi klien, yang menyediakan aliran gambar dan video ke komponen runtime ini untuk mendeteksi anomali menggunakan model pembelajaran mesin.

Anda dapat menggunakan Lookout for Vision Edge Agent API untuk berinteraksi dengan komponen ini dari komponen Greengrass lainnya. API ini diimplementasikan menggunakan gRPC, yang merupakan protokol untuk membuat panggilan prosedur jarak jauh. Untuk informasi selengkapnya, lihat Menulis komponen aplikasi klien dan referensi API Lookout for Vision Edge Agent di Amazon Lookout for Vision Developer Guide.

Untuk informasi selengkapnya tentang cara menggunakan komponen ini, lihat berikut ini:

- Gunakan Amazon Lookout for Vision
- Apa itu Amazon Lookout for Vision? di Panduan Pengembang Lookout for Vision Amazon
- Membuat model Lookout for Vision di Amazon Lookout for Vision Developer Guide.
- Menggunakan model Lookout for Vision pada perangkat edge di Amazon Lookout for Vision Developer Guide.

> **Note**
>
> Komponen Lookout for Vision Edge Agent hanya tersedia dalam hal berikut: Wilayah AWS
>
> - AS Timur (Ohio)
> - AS Timur (Virginia Utara)
> - AS Barat (Oregon)
> - Eropa (Frankfurt)
> - Eropa (Irlandia)
> - Asia Pasifik (Tokyo)

- Asia Pasifik (Seoul)

Topik

Versi

Komponen ini memiliki versi berikut:

- 1.2.x
- 1.1.x
- 1.0.x
- 0.1.x

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Greengrass harus menggunakan arsitektur Armv8 (AArch64) atau x86_64.
- Jika Anda menggunakan versi 1.0.0 atau yang lebih baru dari komponen ini, Python 3.8 atau Python 3.9, termasuk, `pip` diinstal pada perangkat inti Greengrass.

  Jika Anda menggunakan versi 0.1.x dari komponen ini, Python 3.7, termasuk, `pip` diinstal pada perangkat inti Greengrass.

  > **Important**
  >
  > Perangkat harus memiliki salah satu versi Python yang tepat ini. Komponen ini tidak mendukung versi Python yang lebih baru.

- Untuk menggunakan inferensi unit pemrosesan grafis (GPU), perangkat inti harus memenuhi persyaratan berikut. Inferensi GPU bersifat opsional di versi 1.1.0 dan yang lebih baru dari komponen ini.
  - Graphics Processing Unit (GPU) yang mendukung CUDA. Untuk informasi selengkapnya, lihat Memverifikasi Anda Memiliki GPU berkemampuan CUDA di Dokumentasi Toolkit CUDA.
  - cuDNN, CUDA, dan TensorRT diinstal pada perangkat inti Greengrass.
    - Pada perangkat NVIDIA Jetson, seperti Jetson Nano atau Jetson Xavier, cuDNN, CUDA, dan TensorRT diinstal dengan NVIDIA. JetPack Anda tidak perlu melakukan perubahan apa pun. Komponen ini mendukung JetPack 4.4, JetPack4.5, JetPack 4.5.1, dan JetPack4.6.1.

      > **Important**
      >
      > Anda harus menginstal salah satu versi ini JetPack dan bukan versi lain. Layanan Lookout for Vision mengkompilasi model visi komputer untuk platform ini. JetPack

    - Pada perangkat x86 dengan GPU yang memiliki mikroarsitektur NVIDIA Ampere (atau kapasitas komputasi GPU adalah 8.0), lakukan hal berikut:
      - Instal cuDNN dengan mengikuti petunjuk di Panduan Instalasi NVIDIA cuDNN.
      - Instal CUDA versi 11.2 dengan mengikuti petunjuk di Panduan Instalasi NVIDIA CUDA untuk Linux.
      - Instal TensorRT versi 8.2.0 dengan mengikuti petunjuk di Dokumentasi TensorRT NVIDIA.
    - Pada perangkat x86 dengan GPU yang memiliki arsitektur NVIDIA sebelum Ampere (atau kapasitas komputasi GPU kurang dari 8.0), lakukan hal berikut:
      - Instal cuDNN dengan mengikuti petunjuk di Panduan Instalasi NVIDIA cuDNN.

  - Instal CUDA versi 10.2 dengan mengikuti petunjuk di Panduan Instalasi NVIDIA CUDA untuk Linux.
  - Instal TensorRT versi 7.1.3 atau yang lebih baru, tetapi lebih awal dari versi 8.0.0, dengan mengikuti petunjuk di Dokumentasi TensorRT NVIDIA.
- Pengguna sistem yang menjalankan komponen ini harus menjadi anggota grup sistem yang memiliki akses ke GPU pada perangkat. Nama grup ini berbeda menurut sistem operasi. Konsultasikan dokumentasi untuk sistem operasi dan GPU Anda untuk menentukan nama grup sistem ini.

  Misalnya, pada perangkat NVIDIA Jetson, nama grup ini adalah`video`, dan Anda dapat menjalankan perintah berikut untuk menambahkan pengguna sistem ke grup ini. *`Ganti ggc_user`* dengan nama pengguna yang akan ditambahkan.

  ```
  sudo usermod -aG video ggc_user
  ```

Dependensi

Komponen ini tidak memiliki dependensi apa pun.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`Socket`

(Opsional) Soket file tempat Agen Edge beroperasi. Komponen model Lookout for Vision menggunakan soket file ini untuk berkomunikasi dengan Edge Agent. Jika Anda mengubah parameter ini, Anda harus menentukan nilai yang sama saat menerapkan komponen model Lookout for Vision.

Default: `unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock`

File log lokal

Komponen ini menggunakan file log berikut.

```
/greengrass/v2/logs/aws.iot.lookoutvision.EdgeAgent.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* dengan jalur ke folder AWS IoT Greengrass root.

```
sudo tail -f /greengrass/v2/logs/aws.iot.lookoutvision.EdgeAgent.log
```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
1.2.0	Perbaikan bug umum dan perbaikan.
1.1.9	Perbaikan bug umum dan perbaikan.
1.1.8	Perbaikan bug umum dan perbaikan.
1.1.7	Fitur baru • Menginstal `opencv-python-headless` paket di lingkungan virtual Lookout for Vision Edge Agent. Perbaikan bug dan peningkatan • Meningkatkan perhitungan skor kepercayaan diri. • Mengubah ukuran masker model heatmap ke ukuran file asli. • Perbaikan bug umum dan perbaikan.
1.1.6	Fitur baru Menambahkan nilai baru ke `DetectAnomalies` hasilnya. • `anomaly_score` — Angka antara 0,0 dan 1,0 yang menunjukkan seberapa anomali suatu gambar. • `anomaly_threshold` — Ambang batas ditetapkan selama pelatihan model yang menentukan batas antara gambar anomali dan gambar normal.

Versi	Perubahan
	Perbaikan bug umum dan perbaikan.
1.1.4	Fitur baru Ditambahkan dukungan untuk OpenCV untuk mengubah ukuran gambar bila tersedia. Agen Edge menggunakan Pillow saat OpenCV tidak tersedia. Perbaikan bug dan peningkatan Perbaikan bug umum dan perbaikan.
1.1.3	Perbaikan bug umum dan perbaikan.
1.1.1	Perbaikan bug umum dan perbaikan.
1.1.0	Fitur baru • Menambahkan dukungan untuk model segmentasi gambar, yang mengidentifikasi anomali dalam gambar. • Menambahkan dukungan untuk inferensi CPU, sehingga Anda dapat menggunakan model Lookout for Vision pada perangkat inti tanpa GPU. Perbaikan bug dan peningkatan • Perbaikan bug umum dan perbaikan.
1.0.0	Versi komponen Lookout for Vision Edge Agent ini memerlukan versi Python yang berbeda dari versi 0.1.x. Jika Anda ingin meningkatkan dari v0.1.x ke v1.x, Anda harus memutakhirkan instalasi Python pada perangkat inti. Perbaikan bug dan peningkatan • Perbaikan bug umum dan perbaikan.
0.1.37	Perbaikan bug umum dan perbaikan.
0.1.36	Versi awal.

SageMaker Manajer Tepi

> **Important**
>
> SageMaker Edge Manager dihentikan pada 26 April 2024. Untuk informasi selengkapnya tentang melanjutkan penerapan model Anda ke perangkat edge, lihat SageMaker Edge Manager akhir masa pakai.

Komponen Amazon SageMaker Edge Manager (`aws.greengrass.SageMakerEdgeManager`) menginstal biner agen SageMaker Edge Manager.

SageMaker Edge Manager menyediakan manajemen model untuk perangkat edge sehingga Anda dapat mengoptimalkan, mengamankan, memantau, dan memelihara model pembelajaran mesin pada armada perangkat edge. Komponen SageMaker Edge Manager menginstal dan mengelola siklus hidup agen SageMaker Edge Manager di perangkat inti Anda. Anda juga dapat menggunakan SageMaker Edge Manager untuk mengemas dan menggunakan model yang SageMaker dikompilasi NEO sebagai komponen model pada perangkat inti Greengrass. Untuk informasi selengkapnya tentang penggunaan agen SageMaker Edge Manager di perangkat inti Anda, lihatGunakan Amazon SageMaker Edge Manager di perangkat inti Greengrass.

SageMaker Komponen Edge Manager v1.3.x menginstal biner agen Edge Manager v1.20220822.836f3023. Untuk informasi lebih lanjut tentang agen Edge Manager versi biner, lihat Agen Manajer Edge.

> **Note**
>
> Komponen SageMaker Edge Manager hanya tersedia dalam hal berikut Wilayah AWS:
>
> - AS Timur (Ohio)
> - AS Timur (Virginia Utara)
> - US West (Oregon)
> - EU (Frankfurt)
> - EU (Ireland)
> - Asia Pasifik (Tokyo)

Topik

- Versi
- Jenis
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- Berkas log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 1.3.x
- 1.2.x
- 1.1.x
- 1.0.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Greengrass yang berjalan di Amazon Linux 2, platform Linux berbasis Debian (x86_64 atau Armv8), atau Windows (x86_64). Jika Anda tidak memilikinya, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2.
- Python 3.6 atau yang lebih baru, termasuk `pip` untuk versi Python Anda, diinstal pada perangkat inti anda.
- Peran perangkat Greengrass yang dikonfigurasi dengan berikut ini:
  - Hubungan kepercayaan yang memungkinkan `credentials.iot.amazonaws.com` dan `sagemaker.amazonaws.com` untuk meneruskan peran, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

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

  - Kebijakan yang dikelola AmazonSageMakerEdgeDeviceFleetPolicyIAM.
  - Tindakan `s3:PutObject`, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:PutObject"
      ],
```

```
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

- Bucket Amazon S3 yang dibuat sama Akun AWS dan Wilayah AWS sebagai perangkat inti Greengrass Anda. SageMaker Edge Manager memerlukan bucket S3 untuk membuat armada perangkat edge, dan menyimpan data sampel dari inferensi yang sedang berjalan di perangkat Anda. Untuk informasi selengkapnya tentang pembuatan bucket S3, lihat Memulai Amazon S3.
- Armada perangkat SageMaker edge yang menggunakan alias AWS IoT peran yang sama dengan perangkat inti Greengrass Anda. Untuk informasi selengkapnya, lihat Buat armada perangkat edge.
- Perangkat inti Greengrass Anda terdaftar sebagai perangkat tepi di armada perangkat Edge Anda. SageMaker Nama perangkat tepi harus cocok dengan nama AWS IoT benda untuk perangkat inti Anda. Untuk informasi selengkapnya, lihat Daftarkan perangkat inti Greengrass Anda.

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Wajib	Deskripsi
`edge.sagemaker.`*`region`*`.amazonaws.com`	443	Ya	Periksa status pendaftaran perangkat dan kirim metrik ke SageMaker.
`*.s3.amazonaws.com`	443	Ya	Unggah data

Titik Akhir	Port	Wajib	Deskripsi
			tangkapan ke bucket S3 yang Anda tentukan. Anda dapat mengganti * dengan nama setiap bucket tempat Anda mengunggah data.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

1.3.5 and 1.3.6

Tabel berikut mencantumkan dependensi untuk versi 1.3.5 dan 1.3.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

1.3.4

Tabel berikut mencantumkan dependensi untuk versi 1.3.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

1.3.3

Tabel berikut mencantumkan dependensi untuk versi 1.3.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

1.3.2

Tabel berikut mencantumkan dependensi untuk versi 1.3.2 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

1.3.1

Tabel berikut mencantumkan dependensi untuk versi 1.3.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	>=0.0.0	Keras

1.1.1 - 1.3.0

Tabel berikut mencantumkan dependensi untuk versi 1.1.1 - 1.3.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

1.1.0

Tabel berikut mencantumkan dependensi untuk versi 1.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

1.0.3

Tabel berikut mencantumkan dependensi untuk versi 1.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

1.0.1 and 1.0.2

Tabel berikut mencantumkan dependensi untuk versi 1.0.1 dan 1.0.2 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

1.0.0

Tabel berikut mencantumkan dependensi untuk versi 1.0.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

> **Note**
>
> Bagian ini menjelaskan parameter konfigurasi yang Anda tetapkan dalam komponen. Untuk informasi selengkapnya tentang konfigurasi SageMaker Edge Manager terkait, lihat Agen Manajer Edge di Panduan SageMaker Pengembang Amazon.

`DeviceFleetName`

Nama armada perangkat SageMaker Edge Manager yang berisi perangkat inti Greengrass Anda.

Anda harus menentukan nilai untuk parameter ini dalam pembaruan konfigurasi ketika Anda men-deploy komponen ini.

`BucketName`

Nama bucket S3 yang padanya Anda unggah data inferensi yang ditangkap. Nama bucket harus berisi string `sagemaker`.

Jika Anda mengatur `CaptureDataDestination` ke `Cloud`, atau jika Anda mengatur `CaptureDataPeriodicUpload` ke `true`, Anda harus menentukan nilai untuk parameter ini di pembaruan konfigurasi saat Anda men-deploy komponen ini.

> **Note**
>
> Capture data adalah SageMaker fitur yang Anda gunakan untuk mengunggah input inferensi, hasil inferensi, dan data inferensi tambahan ke bucket S3 atau direktori lokal untuk analisis masa depan. Untuk informasi selengkapnya tentang penggunaan data pengambilan dengan SageMaker Edge Manager, lihat Mengelola Model di Panduan SageMaker Pengembang Amazon.

`CaptureDataBatchSize`

(Opsional) Ukuran batch permintaan data tangkapan yang ditangani agen. Nilai ini harus lebih kecil dari ukuran buffer yang Anda tentukan di `CaptureDataBufferSize`. Kami merekomendasikan agar Anda tidak melebihi setengah ukuran buffer.

Agen menangani batch permintaan ketika jumlah permintaan dalam buffer memenuhi jumlah `CaptureDataBatchSize`, atau ketika interval `CaptureDataPushPeriodSeconds` berlalu, mana yang terjadi lebih dahulu.

Default: `10`

`CaptureDataBufferSize`

(Opsional) Jumlah maksimum permintaan data tangkapan yang disimpan dalam buffer.

Default: `30`

`CaptureDataDestination`

(Opsional) Tujuan di mana Anda menyimpan data yang diambil. Parameter ini dapat memiliki nilai berikut:

- `Cloud`—Mengunggah data yang ditangkap ke bucket S3 yang Anda tentukan di `BucketName`.
- `Disk`—Menuliskan data yang ditangkap pada direktori kerja komponen.

Jika Anda menentukan `Disk`, Anda juga dapat memilih untuk mengunggah data yang diambil secara berkala ke bucket S3 dengan menetapkan `CaptureDataPeriodicUpload` ke `true`.

Default: `Cloud`

`CaptureDataPeriodicUpload`

(Opsional) Nilai string yang menentukan apakah akan secara berkala meng-upload data yang ditangkap. Nilai yang didukung adalah `true` dan `false`.

Atur parameter ini ke `true` jika Anda mengatur `CaptureDataDestination` ke `Disk`, dan Anda juga ingin agen untuk secara berkala meng-upload data yang diambil bucket S3 Anda.

Default: `false`

`CaptureDataPeriodicUploadPeriodSeconds`

(Opsional) Interval dalam hitungan detik saat agen SageMaker Edge Manager mengunggah data yang diambil ke bucket S3. Gunakan parameter ini jika Anda mengatur `CaptureDataPeriodicUpload` ke `true`.

Default: `8`

`CaptureDataPushPeriodSeconds`

(Opsional) Interval dalam hitungan detik di mana agen SageMaker Edge Manager menangani sekumpulan permintaan data pengambilan dari buffer.

Agen menangani batch permintaan ketika jumlah permintaan dalam buffer memenuhi jumlah `CaptureDataBatchSize`, atau ketika interval `CaptureDataPushPeriodSeconds` berlalu, mana yang terjadi lebih dahulu.

Default: `4`

`CaptureDataBase64EmbedLimit`

(Opsional) Ukuran maksimum dalam byte data yang diambil yang diunggah agen SageMaker Edge Manager.

Default: `3072`

`FolderPrefix`

(Opsional) Nama folder tempat agen menulis data yang ditangkap. Jika Anda mengatur `CaptureDataDestination` ke`Disk`, agen membuat folder di direktori yang ditentukan oleh`CaptureDataDiskPath`. Jika Anda mengatur `CaptureDataDestination` ke `Cloud`, atau

jika Anda mengatur `CaptureDataPeriodicUpload` ke `true`, agen akan membuat folder di bucket S3 Anda.

Default: `sme-capture`

`CaptureDataDiskPath`

Fitur ini tersedia di v1.1.0 dan versi yang lebih baru dari komponen SageMaker Edge Manager.

(Opsional) Jalur ke folder tempat agen membuat folder data yang diambil. Jika Anda menyetel `CaptureDataDestination` ke`Disk`, agen akan membuat folder data yang diambil di direktori ini. Jika Anda tidak menentukan nilai ini, agen akan membuat folder data yang diambil di direktori kerja komponen. Gunakan `FolderPrefix` parameter untuk menentukan nama folder data yang diambil.

Default: *`/greengrass/v2`*`/work/aws.greengrass.SageMakerEdgeManager/capture`

`LocalDataRootPath`

Fitur ini tersedia di v1.2.0 dan versi yang lebih baru dari komponen SageMaker Edge Manager.

(Opsional) Jalur tempat komponen ini menyimpan data berikut pada perangkat inti:

- Database lokal untuk data runtime saat Anda menyetel `DbEnable` ke`true`.
- SageMaker Model yang dikompilasi neo yang diunduh komponen ini secara otomatis saat Anda `DeploymentEnable` menyetelnya. `true`

Default: *`/greengrass/v2`*`/work/aws.greengrass.SageMakerEdgeManager`

`DbEnable`

(Opsional) Anda dapat mengaktifkan komponen ini untuk menyimpan data runtime dalam database lokal untuk menyimpan data, jika komponen gagal atau perangkat kehilangan daya.

Database ini membutuhkan 5 MB penyimpanan pada sistem file perangkat inti.

Default: `false`

`DeploymentEnable`

Fitur ini tersedia di v1.2.0 dan versi yang lebih baru dari komponen SageMaker Edge Manager.

(Opsional) Anda dapat mengaktifkan komponen ini untuk secara otomatis mengambil model yang SageMaker dikompilasi NEO dari yang Anda unggah ke Amazon S3. Setelah Anda mengunggah model baru ke Amazon S3, gunakan SageMaker Studio atau SageMaker API untuk menerapkan

model baru ke perangkat inti ini. Saat mengaktifkan fitur ini, Anda dapat menerapkan model baru ke perangkat inti tanpa perlu membuat AWS IoT Greengrass penerapan.

> **Important**
>
> Untuk menggunakan fitur ini, Anda harus mengatur `DbEnable` ke`true`. Fitur ini menggunakan database lokal untuk melacak model yang diambil dari file. AWS Cloud

Default: `false`

`DeploymentPollInterval`

Fitur ini tersedia di v1.2.0 dan versi yang lebih baru dari komponen SageMaker Edge Manager.

(Opsional) Jumlah waktu (dalam menit) di mana komponen ini memeriksa model baru untuk diunduh. Opsi ini berlaku saat Anda menyetel `DeploymentEnable` ke`true`.

Default: `1440` (1 hari)

`DLRBackendOptions`

Fitur ini tersedia di v1.2.0 dan versi yang lebih baru dari komponen SageMaker Edge Manager.

(Opsional) Bendera runtime DLR untuk disetel di runtime DLR yang digunakan komponen ini. Anda dapat mengatur bendera berikut:

- `TVM_TENSORRT_CACHE_DIR`- Aktifkan caching model TensorRT. Tentukan jalur absolut ke folder yang ada yang memiliki izin baca/tulis.
- `TVM_TENSORRT_CACHE_DISK_SIZE_MB`— Menetapkan batas atas folder cache model TensorRT. Ketika ukuran direktori tumbuh melampaui batas ini, mesin cache yang paling sedikit digunakan dihapus. Nilai defaultnya adalah 512 MB.

Misalnya, Anda dapat menyetel parameter ini ke nilai berikut untuk mengaktifkan caching model TensorRT dan membatasi ukuran cache hingga 800 MB.

```
TVM_TENSORRT_CACHE_DIR=/data/secured_folder/trt/cache;
 TVM_TENSORRT_CACHE_DISK_SIZE_MB=800
```

`SagemakerEdgeLogVerbose`

(Opsional) Nilai string yang menentukan apakah akan mengaktifkan pencatatan debug. Nilai yang didukung adalah `true` dan `false`.

Default: `false`

`UnixSocketName`

(Opsional) Lokasi deskriptor file soket SageMaker Edge Manager pada perangkat inti.

Default: `/tmp/aws.greengrass.SageMakerEdgeManager.sock`

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan bahwa perangkat inti adalah bagian dari *MyEdgeDeviceFleet*dan bahwa agen menulis data pengambilan baik ke perangkat dan ke bucket S3. Konfigurasi ini juga memungkinkan pencatatan debug.

```
{
    "DeviceFleetName": "MyEdgeDeviceFleet",
    "BucketName": "DOC-EXAMPLE-BUCKET",
    "CaptureDataDestination": "Disk",
    "CaptureDataPeriodicUpload": "true",
    "SagemakerEdgeLogVerbose": "true"
}
```

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.greengrass.SageMakerEdgeManager.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.SageMakerEdgeManager.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

Linux

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

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs\aws.greengrass.SageMakerEdgeManager.log -Tail
 10 -Wait
```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
1.3.6	Versi diperbarui untuk Greengrass nucleus 2.12.5 rilis.
1.3.5	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
1.3.4	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
1.3.3	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
1.3.2	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
1.3.1	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
1.3.0	Fitur baru • Menambahkan dukungan untuk manajemen ukuran disk cache TensorRT. • Menambahkan `TVM_TENSORRT_CACHE_DISK_SIZE_MB` flag opsional ke BackendOptions parameter DLR untuk mengatur batas ukuran untuk model cache pada disk. Perbaikan • Memberikan konkurensi prediksi yang lebih baik. Ini membantu untuk mendapatkan penggunaan mesin akselerator perangkat yang lebih baik, seperti GPU.

Versi	Perubahan
1.2.0	Fitur baru • Menambahkan dukungan untuk komponen ini untuk secara otomatis mengambil model yang SageMaker dikompilasi NEO yang Anda unggah ke Amazon S3. Saat mengaktifkan fitur ini, Anda dapat menerapkan model baru ke perangkat inti tanpa perlu membuat AWS IoT Greengrass penerapan. • Menambahkan dukungan untuk database cadangan yang digunakan komponen ini untuk menyimpan data runtime, jika komponen gagal atau perangkat kehilangan daya. • Menambahkan dukungan bagi Anda untuk mengonfigurasi flag runtime DLR saat Anda mengonfigurasi komponen ini.
1.1.1	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
1.1.0	Fitur baru • Menambahkan dukungan untuk perangkat inti Greengrass yang menjalankan Amazon Linux 2. • Menambahkan parameter `CaptureDataDiskPath` konfigurasi baru. Anda dapat menggunakan parameter ini untuk menentukan jalur folder data yang diambil pada perangkat Anda. Perbaikan bug dan peningkatan • Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
1.0.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
1.0.2	Perbaikan bug dan peningkatan Memperbarui skrip instalasi dalam siklus hidup komponen. Perangkat inti Anda sekarang harus memiliki Python 3.6 atau yang lebih baru, termasuk `pip` untuk versi Python Anda, yang diinstal pada perangkat sebelum Anda men-deploy komponen ini.
1.0.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
1.0.0	Versi awal.

Klasifikasi citra DLR

Komponen klasifikasi citra DLR (`aws.greengrass.DLRImageClassification`) berisi contoh kode inferensi untuk melakukan inferensi klasifikasi gambar menggunakan Deep Learning Runtime dan model resnet-50. Komponen ini menggunakan varian Penyimpanan model klasifikasi gambar DLR dan komponen Runtime DLR sebagai dependensi untuk mengunduh DLR dan model sampel.

Untuk menggunakan komponen inferensi ini dengan model DLR yang terlatih khusus, buat versi kustom dari komponen penyimpanan model dependen. Untuk menggunakan kode inferensi kustom Anda sendiri, Anda dapat menggunakan resep komponen ini sebagai templat untuk membuat komponen inferensi kustom.

Topik

- Versi
- Jenis
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- File log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.1.x
- 2.0.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat.
- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```
- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

    ```
    pip3 install --upgrade numpy
    ```
  - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

    Untuk mengaktifkan tumpukan kamera lama

    1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

       ```
       sudo raspi-config
       ```
    2. Pilih Opsi Antarmuka.
    3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.

4. Reboot Raspberry Pi.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.13 and 2.1.14

Tabel berikut mencantumkan dependensi untuk versi 2.1.13 dan 2.1.14 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.12

Tabel berikut mencantumkan dependensi untuk versi 2.1.12 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.11

Tabel berikut mencantumkan dependensi untuk versi 2.1.11 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.10

Tabel berikut mencantumkan dependensi untuk versi 2.1.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.9

Tabel berikut mencantumkan dependensi untuk versi 2.1.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.4 - 2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 hingga 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak
Toko model klasifikasi gambar DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.0.x

Tabel berikut mencantumkan dependensi untuk versi 2.0.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	~2.0.0	Lunak
Penyimpanan model klasifikasi gambar DLR	~2.0.0	Keras
DLR	~1.3.0	Lunak

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

2.1.x

`accessControl`

(Opsional) Objek yang berisi kebijakan otorisasi yang memungkinkan komponen untuk mempublikasikan pesan ke topik pemberitahuan default.

Default:

```
{
   "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.DLRImageClassification:mqttproxy:1": {
         "policyDescription": "Allows access to publish via topic ml/dlr/image-
classification.",
         "operations": [
            "aws.greengrass#PublishToIoTCore"
         ],
         "resources": [
            "ml/dlr/image-classification"
         ]
      }
   }
}
```

`PublishResultsOnTopic`

(Opsional) Topik di mana Anda ingin mempublikasikan hasil inferensi. Jika Anda mengubah nilai ini, maka Anda juga harus mengubah nilai `resources` di parameter `accessControl` agar cocok dengan nama topik kustom Anda.

Default: `ml/dlr/image-classification`

`Accelerator`

Akselerator yang ingin Anda gunakan. Nilai yang didukung adalah `cpu` dan `gpu`.

Model sampel dalam komponen model dependen hanya mendukung akselerasi CPU. Untuk menggunakan akselerasi GPU dengan model kustom yang berbeda, buat komponen model kustom untuk menimpa komponen model publik.

Default: `cpu`

`ImageDirectory`

(Opsional) Jalur folder pada perangkat di mana komponen inferensi membaca gambar. Anda dapat mengubah nilai ini ke lokasi mana pun di perangkat Anda yang memiliki akses baca/tulis.

Default: */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/image_classification/sample_images/`

> **Note**
>
> Jika Anda menetapkan nilai `UseCamera` ke `true`, maka parameter konfigurasi ini diabaikan.

`ImageName`

(Opsional) Nama gambar yang digunakan oleh komponen inferensi sebagai masukan untuk membuat prediksi. Komponen mencari gambar dalam folder yang ditentukan dalam `ImageDirectory`. Secara default, komponen menggunakan gambar sampel di direktori gambar default. AWS IoT Greengrass mendukung format gambar berikut:`jpeg`,`jpg`,`png`, dannpy.

Default: `cat.jpeg`

> **Note**
>
> Jika Anda menetapkan nilai `UseCamera` ke `true`, maka parameter konfigurasi ini diabaikan.

`InferenceInterval`

(Opsional) Waktu dalam detik antara setiap prediksi yang dibuat oleh kode inferensi. Kode inferensi sampel berjalan tanpa batas waktu dan mengulangi prediksinya pada interval waktu yang ditentukan. Misalnya, Anda dapat mengubahnya menjadi interval yang lebih pendek jika ingin menggunakan gambar yang diambil oleh kamera untuk prediksi real-time.

Default: `3600`

`ModelResourceKey`

(Opsional) Model yang digunakan dalam komponen model publik dependen. Ubah parameter ini hanya jika Anda menimpa komponen model publik dengan komponen kustom.

Default:

```
{
    "armv7l": "DLR-resnet50-armv7l-cpu-ImageClassification",
    "aarch64": "DLR-resnet50-aarch64-cpu-ImageClassification",
```

```
    "x86_64": "DLR-resnet50-x86_64-cpu-ImageClassification",
    "windows": "DLR-resnet50-win-cpu-ImageClassification"
}
```

`UseCamera`

(Opsional) Nilai string yang menentukan apakah akan menggunakan gambar dari kamera yang terhubung ke perangkat inti Greengrass. Nilai yang didukung adalah `true` dan `false`.

Ketika Anda menetapkan nilai ini ke `true`, kode kesimpulan sampel akan mengakses kamera pada perangkat Anda dan menjalankan kesimpulan secara lokal pada gambar yang ditangkap. Nilai `ImageName` dan parameter `ImageDirectory` diabaikan. Pastikan bahwa pengguna yang menjalankan komponen ini memiliki akses baca/tulis ke lokasi di mana kamera menyimpan gambar yang diambil.

Default: `false`

> **Note**
>
> Ketika Anda melihat resep komponen ini, parameter konfigurasi `UseCamera` tidak muncul dalam konfigurasi default. Namun, Anda dapat mengubah nilai parameter ini dalam pembaruan gabungan konfigurasi saat Anda men-deploy komponen.
> Ketika Anda mengatur `UseCamera` ke `true`, Anda juga harus membuat symlink untuk mengaktifkan komponen inferensi untuk mengakses kamera Anda dari lingkungan virtual yang dibuat oleh komponen waktu aktif. Untuk informasi lebih lanjut tentang penggunaan kamera dengan komponen inferensi sampel, lihat Perbarui konfigurasi komponen.

2.0.x

`MLRootPath`

(Opsional) Jalur folder pada perangkat inti Linux tempat komponen inferensi membaca gambar dan menulis hasil inferensi. Anda dapat mengubah nilai ini ke lokasi mana pun di perangkat Anda yang padanya pengguna menjalankan komponen ini memiliki akses baca/tulis.

Default: */greengrass/v2*`/work/variant.DLR/greengrass_ml`

Default: */greengrass/v2*`/work/variant.TensorFlowLite/greengrass_ml`

`Accelerator`

Akselerator yang ingin Anda gunakan. Nilai yang didukung adalah `cpu` dan `gpu`.

Model sampel dalam komponen model dependen hanya mendukung akselerasi CPU. Untuk menggunakan akselerasi GPU dengan model kustom yang berbeda, buat komponen model kustom untuk menimpa komponen model publik.

Default: `cpu`

`ImageName`

(Opsional) Nama gambar yang digunakan oleh komponen inferensi sebagai masukan untuk membuat prediksi. Komponen mencari gambar dalam folder yang ditentukan dalam `ImageDirectory`. Lokasi defaultnya adalah*`MLRootPath`*`/images`. AWS IoT Greengrass mendukung format gambar berikut:`jpeg`,`jpg`,`png`, dan`npy`.

Default: `cat.jpeg`

`InferenceInterval`

(Opsional) Waktu dalam detik antara setiap prediksi yang dibuat oleh kode inferensi. Kode inferensi sampel berjalan tanpa batas waktu dan mengulangi prediksinya pada interval waktu yang ditentukan. Misalnya, Anda dapat mengubahnya menjadi interval yang lebih pendek jika ingin menggunakan gambar yang diambil oleh kamera untuk prediksi real-time.

Default: `3600`

`ModelResourceKey`

(Opsional) Model yang digunakan dalam komponen model publik dependen. Ubah parameter ini hanya jika Anda menimpa komponen model publik dengan komponen kustom.

Default:

```
armv7l: "DLR-resnet50-armv7l-cpu-ImageClassification"
x86_64: "DLR-resnet50-x86_64-cpu-ImageClassification"
```

File log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.greengrass.DLRImageClassification.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.DLRImageClassification.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.DLRImageClassification.log -
  Tail 10 -Wait
  ```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.14	Versi diperbarui untuk Greengrass nucleus 2.12.5 rilis.
2.1.13	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.12	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.11	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.10	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.

Versi	Perubahan
2.1.9	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.5	Komponen dirilis di semua Wilayah AWS.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis. Versi ini tidak tersedia di Eropa (London) (`eu-west-2`).
2.1.3	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.

Versi	Perubahan
2.1.1	Fitur baru • Gunakan Deep Learning Runtime v1.6.0. • Tambahkan dukungan untuk klasifikasi gambar sampel pada platform Armv8 (AArch64). Hal ini akan memperluas dukungan machine learning untuk perangkat inti Greengrass yang menjalankan NVIDIA Jetson, seperti Jetson Nano. • Aktifkan integrasi kamera untuk inferensi sampel. Gunakan parameter konfigurasi `UseCamera` yang baru untuk mengaktifkan kode kesimpulan sampel untuk mengakses kamera pada perangkat inti Greengrass Anda dan jalankan inferensi lokal pada gambar yang ditangkap. • Tambahkan dukungan untuk menerbitkan hasil inferensi ke AWS Cloud. Gunakan parameter konfigurasi `PublishResultsOnTopic` yang baru untuk menentukan topik di mana Anda ingin mempublikasikan hasil. • Tambahkan parameter konfigurasi `ImageDirectory` yang baru yang memungkinkan Anda untuk menentukan direktori kustom untuk gambar di mana Anda ingin melakukan inferensi. Perbaikan bug dan peningkatan • Tulis hasil inferensi ke file log komponen dan bukan file inferensi terpisah. • Gunakan modul logging perangkat lunak AWS IoT Greengrass inti untuk mencatat output komponen. • Gunakan AWS IoT Device SDK untuk membaca konfigurasi komponen dan menerapkan perubahan konfigurasi.
2.0.4	Versi awal.

Deteksi objek DLR

Komponen deteksi objek DLR (`aws.greengrass.DLRObjectDetection`) berisi contoh kode inferensi untuk melakukan inferensi deteksi objek dengan menggunakan Deep Learning Runtime dan sampel model yang terlatih sebelumnya. Komponen ini menggunakan varian Penyimpanan model

deteksi DLR dan komponen Runtime DLR sebagai dependensi untuk mengunduh DLR dan model sampel.

Untuk menggunakan komponen inferensi ini dengan model DLR yang terlatih khusus, buat versi kustom dari komponen penyimpanan model dependen. Untuk menggunakan kode inferensi kustom Anda sendiri, Anda dapat menggunakan resep komponen ini sebagai templat untuk membuat komponen inferensi kustom.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.1.x
- 2.0.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat.
- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```
- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

    ```
    pip3 install --upgrade numpy
    ```
  - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

    Untuk mengaktifkan tumpukan kamera lama

    1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

       ```
       sudo raspi-config
       ```
    2. Pilih Opsi Antarmuka.
    3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.
    4. Reboot Raspberry Pi.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi

yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.13 and 2.1.14

Tabel berikut mencantumkan dependensi untuk versi 2.1.13 dan 2.1.14 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.12

Tabel berikut mencantumkan dependensi untuk versi 2.1.12 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.11

Tabel berikut mencantumkan dependensi untuk versi 2.1.11 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak

Dependensi	Versi yang kompatibel	Jenis dependensi
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.10

Tabel berikut mencantumkan dependensi untuk versi 2.1.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.9

Tabel berikut mencantumkan dependensi untuk versi 2.1.9 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.4 - 2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 hingga 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak
Toko model deteksi objek DLR	~2.1.0	Keras
DLR	~1.6.0	Keras

2.0.x

Tabel berikut mencantumkan dependensi untuk versi 2.0.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	~2.0.0	Lunak
Penyimpanan model deteksi DLR	~2.0.0	Keras
DLR	~1.3.0	Lunak

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

2.1.x

`accessControl`

(Opsional) Objek yang berisi kebijakan otorisasi yang memungkinkan komponen untuk mempublikasikan pesan ke topik pemberitahuan default.

Default:

```
{
   "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.DLRObjectDetection:mqttproxy:1": {
```

```
         "policyDescription": "Allows access to publish via topic ml/dlr/object-
detection.",
         "operations": [
            "aws.greengrass#PublishToIoTCore"
         ],
         "resources": [
            "ml/dlr/object-detection"
         ]
      }
   }
}
```

`PublishResultsOnTopic`

(Opsional) Topik di mana Anda ingin mempublikasikan hasil inferensi. Jika Anda mengubah nilai ini, maka Anda juga harus mengubah nilai `resources` di parameter `accessControl` agar cocok dengan nama topik kustom Anda.

Default: `ml/dlr/object-detection`

`Accelerator`

Akselerator yang ingin Anda gunakan. Nilai yang didukung adalah `cpu` dan `gpu`.

Model sampel dalam komponen model dependen hanya mendukung akselerasi CPU. Untuk menggunakan akselerasi GPU dengan model kustom yang berbeda, buat komponen model kustom untuk menimpa komponen model publik.

Default: `cpu`

`ImageDirectory`

(Opsional) Jalur folder pada perangkat di mana komponen inferensi membaca gambar. Anda dapat mengubah nilai ini ke lokasi mana pun di perangkat Anda yang memiliki akses baca/tulis.

Default: *`/greengrass/v2`*`/packages/artifacts-unarchived/`*`component-name`*`/object_detection/sample_images/`

> **Note**
>
> Jika Anda menetapkan nilai `UseCamera` ke `true`, maka parameter konfigurasi ini diabaikan.

`ImageName`

(Opsional) Nama gambar yang digunakan oleh komponen inferensi sebagai masukan untuk membuat prediksi. Komponen mencari gambar dalam folder yang ditentukan dalam `ImageDirectory`. Secara default, komponen menggunakan gambar sampel di direktori gambar default. AWS IoT Greengrass mendukung format gambar berikut:`jpeg`,`jpg`,`png`, dannpy.

Default: `objects.jpg`

> **Note**
>
> Jika Anda menetapkan nilai `UseCamera` ke `true`, maka parameter konfigurasi ini diabaikan.

`InferenceInterval`

(Opsional) Waktu dalam detik antara setiap prediksi yang dibuat oleh kode inferensi. Kode inferensi sampel berjalan tanpa batas waktu dan mengulangi prediksinya pada interval waktu yang ditentukan. Misalnya, Anda dapat mengubahnya menjadi interval yang lebih pendek jika ingin menggunakan gambar yang diambil oleh kamera untuk prediksi real-time.

Default: `3600`

`ModelResourceKey`

(Opsional) Model yang digunakan dalam komponen model publik dependen. Ubah parameter ini hanya jika Anda menimpa komponen model publik dengan komponen kustom.

Default:

```
{
    "armv7l": "DLR-yolo3-armv7l-cpu-ObjectDetection",
    "aarch64": "DLR-yolo3-aarch64-gpu-ObjectDetection",
    "x86_64": "DLR-yolo3-x86_64-cpu-ObjectDetection",
    "windows": "DLR-resnet50-win-cpu-ObjectDetection"
}
```

`UseCamera`

(Opsional) Nilai string yang menentukan apakah akan menggunakan gambar dari kamera yang terhubung ke perangkat inti Greengrass. Nilai yang didukung adalah `true` dan `false`.

Ketika Anda menetapkan nilai ini ke `true`, kode kesimpulan sampel akan mengakses kamera pada perangkat Anda dan menjalankan kesimpulan secara lokal pada gambar yang ditangkap. Nilai `ImageName` dan parameter `ImageDirectory` diabaikan. Pastikan bahwa pengguna yang menjalankan komponen ini memiliki akses baca/tulis ke lokasi di mana kamera menyimpan gambar yang diambil.

Default: `false`

> **Note**
>
> Ketika Anda melihat resep komponen ini, parameter konfigurasi `UseCamera` tidak muncul dalam konfigurasi default. Namun, Anda dapat mengubah nilai parameter ini dalam pembaruan gabungan konfigurasi saat Anda men-deploy komponen.
> Ketika Anda mengatur `UseCamera` ke `true`, Anda juga harus membuat symlink untuk mengaktifkan komponen inferensi untuk mengakses kamera Anda dari lingkungan virtual yang dibuat oleh komponen waktu aktif. Untuk informasi lebih lanjut tentang penggunaan kamera dengan komponen inferensi sampel, lihat Perbarui konfigurasi komponen.

2.0.x

`MLRootPath`

(Opsional) Jalur folder pada perangkat inti Linux tempat komponen inferensi membaca gambar dan menulis hasil inferensi. Anda dapat mengubah nilai ini ke lokasi mana pun di perangkat Anda yang padanya pengguna menjalankan komponen ini memiliki akses baca/tulis.

Default: `/greengrass/v2/work/variant.DLR/greengrass_ml`

Default: `/greengrass/v2/work/variant.TensorFlowLite/greengrass_ml`

`Accelerator`

Jangan diubah. Saat ini, satu-satunya nilai yang didukung untuk akselerator adalah `cpu`, karena model dalam komponen model dependen dikompilasi hanya untuk akselerator CPU.

`ImageName`

(Opsional) Nama gambar yang digunakan oleh komponen inferensi sebagai masukan untuk membuat prediksi. Komponen mencari gambar dalam folder yang ditentukan dalam `ImageDirectory`. Lokasi defaultnya adalah*`MLRootPath`*`/images`. AWS IoT Greengrass mendukung format gambar berikut:`jpeg`,`jpg`,`png`, dan`npy`.

Default: `objects.jpg`

`InferenceInterval`

(Opsional) Waktu dalam detik antara setiap prediksi yang dibuat oleh kode inferensi. Kode inferensi sampel berjalan tanpa batas waktu dan mengulangi prediksinya pada interval waktu yang ditentukan. Misalnya, Anda dapat mengubahnya menjadi interval yang lebih pendek jika ingin menggunakan gambar yang diambil oleh kamera untuk prediksi real-time.

Default: `3600`

`ModelResourceKey`

(Opsional) Model yang digunakan dalam komponen model publik dependen. Ubah parameter ini hanya jika Anda menimpa komponen model publik dengan komponen kustom.

Default:

```
{
    armv7l: "DLR-yolo3-armv7l-cpu-ObjectDetection",
    x86_64: "DLR-yolo3-x86_64-cpu-ObjectDetection"
}
```

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.greengrass.DLRObjectDetection.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.DLRObjectDetection.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.DLRObjectDetection.log -Tail 10
   -Wait
  ```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.14	Versi diperbarui untuk Greengrass nucleus 2.12.5 rilis.
2.1.13	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.12	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.11	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.10	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.9	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.5	Komponen dirilis di semua Wilayah AWS.

Versi	Perubahan
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis. Versi ini tidak tersedia di Eropa (London) (`eu-west-2`).
2.1.3	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.2	Perbaikan bug dan peningkatan • Memperbaiki masalah penskalaan gambar yang mengakibatkan kotak batas yang tidak akurat dalam hasil inferensi deteksi objek DLR sampel.

Versi	Perubahan
2.1.1	Fitur baru • Gunakan Deep Learning Runtime v1.6.0. • Tambahkan dukungan untuk deteksi objek sampel pada platform Armv8 (AArch64). Hal ini akan memperluas dukungan machine learning untuk perangkat inti Greengrass yang menjalankan NVIDIA Jetson, seperti Jetson Nano. • Aktifkan integrasi kamera untuk inferensi sampel. Gunakan parameter konfigurasi `UseCamera` yang baru untuk mengaktifkan kode kesimpulan sampel untuk mengakses kamera pada perangkat inti Greengrass Anda dan jalankan inferensi lokal pada gambar yang ditangkap. • Tambahkan dukungan untuk menerbitkan hasil inferensi ke AWS Cloud. Gunakan parameter konfigurasi `PublishResultsOnTopic` yang baru untuk menentukan topik di mana Anda ingin mempublikasikan hasil. • Tambahkan parameter konfigurasi `ImageDirectory` yang baru yang memungkinkan Anda untuk menentukan direktori kustom untuk gambar di mana Anda ingin melakukan inferensi. Perbaikan bug dan peningkatan • Tulis hasil inferensi ke file log komponen dan bukan file inferensi terpisah. • Gunakan modul logging perangkat lunak AWS IoT Greengrass inti untuk mencatat output komponen. • Gunakan AWS IoT Device SDK untuk membaca konfigurasi komponen dan menerapkan perubahan konfigurasi.
2.0.4	Versi awal.

Penyimpanan model klasifikasi gambar DLR

Toko model klasifikasi gambar DLR adalah komponen model pembelajaran mesin yang berisi ResNet model -50 yang telah dilatih sebelumnya sebagai artefak Greengrass. Model pra-terlatih yang digunakan dalam komponen ini diambil dari Kebun Binatang Model GluonCV dan dikompilasi menggunakan SageMaker Neo Deep Learning Runtime.

Komponen inferensi klasifikasi citra DLR menggunakan komponen ini sebagai dependensi untuk sumber model. Untuk menggunakan model DLR yang terlatih khusus, buat versi kustomisasi komponen model ini, dan sertakan model kustom Anda sebagai artefak komponen. Anda dapat menggunakan resep komponen ini sebagai templat untuk membuat komponen model kustom.

> **Note**
>
> Nama komponen penyimpanan model klasifikasi gambar DLR bervariasi tergantung pada versinya. Nama komponen untuk versi 2.1.x dan versi yang lebih baru adalah `variant.DLR.ImageClassification.ModelStore`. Nama komponen untuk versi 2.0.x adalah `variant.ImageClassification.ModelStore`.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.1.x (`variant.DLR.ImageClassification.ModelStore`)
- 2.0.x (`variant.ImageClassification.ModelStore`)

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat.
- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```
- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

    ```
    pip3 install --upgrade numpy
    ```
  - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

    Untuk mengaktifkan tumpukan kamera lama

    1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

       ```
       sudo raspi-config
       ```
    2. Pilih Opsi Antarmuka.
    3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.

4. Reboot Raspberry Pi.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.12 - 2.1.14

Tabel berikut mencantumkan dependensi untuk versi 2.1.12 dan 2.1.13 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak

2.1.11

Tabel berikut mencantumkan dependensi untuk versi 2.1.11 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak

2.1.10

Tabel berikut mencantumkan dependensi untuk versi 2.1.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak

2.1.9

Tabel berikut mencantumkan dependensi untuk versi 2.1.9 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak

2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak

2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak

2.0.x

Tabel berikut mencantumkan dependensi untuk versi 2.0.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	~2.0.0	Lunak

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

File log lokal

Komponen ini tidak mengeluarkan log.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.13	Versi diperbarui untuk Greengrass nucleus 2.12.5 rilis.
2.1.12	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.11	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.10	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.9	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.5	Fitur baru • Menambahkan contoh model klasifikasi gambar untuk perangkat inti Windows.

Versi	Perubahan
	• Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.1.3	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.1.1	Fitur baru • Tambahkan contoh model klasifikasi gambar ResNet -50 untuk platform Armv8 (AArch64). Hal ini akan memperluas dukungan machine learning untuk perangkat inti Greengrass yang menjalankan NVIDIA Jetson, seperti Jetson Nano.
2.0.4	Versi awal.

Penyimpanan model deteksi DLR

Penyimpanan model deteksi objek DLR adalah komponen model machine learning yang berisi model YOLOv3 yang terlatih sebelumnya sebagai artefak Greengrass. Model sampel yang digunakan dalam komponen ini diambil dari Kebun Binatang Model GluonCV dan dikompilasi menggunakan SageMaker Neo Deep Learning Runtime.

Komponen inferensi deteksi citra DLR menggunakan komponen ini sebagai dependensi untuk sumber model. Untuk menggunakan model DLR yang terlatih khusus, buat versi kustomisasi komponen model ini, dan sertakan model kustom Anda sebagai artefak komponen. Anda dapat menggunakan resep komponen ini sebagai templat untuk membuat komponen model kustom.

> **Note**
>
> Nama komponen penyimpanan model deteksi objek DLR bervariasi tergantung pada versinya. Nama komponen untuk versi 2.1.x dan versi yang lebih baru adalah `variant.DLR.ObjectDetection.ModelStore`. Nama komponen untuk versi 2.0.x adalah `variant.ObjectDetection.ModelStore`.

Topik

- Versi
- Jenis
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- Berkas log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.1.x
- 2.0.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat.

- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

    ```
    pip3 install --upgrade numpy
    ```

  - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

    Untuk mengaktifkan tumpukan kamera lama

    1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Pilih Opsi Antarmuka.
    3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.
    4. Reboot Raspberry Pi.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.13 and 2.1.14

Tabel berikut mencantumkan dependensi untuk versi 2.1.13 dan 2.1.14 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak

2.1.12

Tabel berikut mencantumkan dependensi untuk versi 2.1.12 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak

2.1.11

Tabel berikut mencantumkan dependensi untuk versi 2.1.11 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak

2.1.10

Tabel berikut mencantumkan dependensi untuk versi 2.1.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak

2.1.9

Tabel berikut mencantumkan dependensi untuk versi 2.1.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak

2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak

2.1.5 and 2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dan 2.1.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak

2.0.x

Tabel berikut mencantumkan dependensi untuk versi 2.0.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	~2.0.0	Lunak

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

Berkas log lokal

Komponen ini tidak mengeluarkan log.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.14	Versi diperbarui untuk Greengrass nucleus 2.12.5 rilis.
2.1.13	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.12	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.11	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.10	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.9	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.6	Menambahkan model CPU untuk memperbaiki masalah pada perangkat Armv8 (AArch64).
2.1.5	Fitur baru • Menambahkan contoh model deteksi objek untuk perangkat inti Windows. Perbaikan bug dan peningkatan • Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.1.3	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.

Versi	Perubahan
2.1.1	Fitur baru • Tambahkan sampel model deteksi objek YOLOv3 pada platform Armv8 (AArch64). Hal ini akan memperluas dukungan machine learning untuk perangkat inti Greengrass yang menjalankan NVIDIA Jetson, seperti Jetson Nano.
2.0.4	Versi awal.

Runtime DLR

Komponen runtime DLR (`variant.DLR`) berisi skrip yang menginstal Deep Learning Runtime (DLR) dan dependensinya di lingkungan virtual di perangkat Anda. Komponen Klasifikasi citra DLR dan Deteksi objek DLR menggunakan komponen ini sebagai dependensi untuk menginstal DLR. Versi komponen 1.6.x menginstal DLR v1.6.0 dan versi komponen 1.3.x menginstal DLR v1.3.0.

Untuk menggunakan runtime yang berbeda, Anda dapat menggunakan resep komponen ini sebagai template untuk membuat komponen pembelajaran mesin kustom.

Topik

Versi

Komponen ini memiliki versi berikut:

- 1.6.x

- 1.3.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat.
- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```
- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

    ```
    pip3 install --upgrade numpy
    ```
  - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

Untuk mengaktifkan tumpukan kamera lama

1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

   ```
   sudo raspi-config
   ```

2. Pilih Opsi Antarmuka.
3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.
4. Reboot Raspberry Pi.

Titik akhir dan port

Secara default, komponen ini menggunakan skrip installer untuk menginstal paket menggunakan`apt`,, `yum`, `brew`, dan `pip` perintah, tergantung pada platform apa yang digunakan perangkat inti. Komponen ini harus dapat melakukan permintaan keluar ke berbagai indeks paket dan repositori untuk menjalankan skrip installer. Untuk mengizinkan lalu lintas keluar komponen ini melalui proxy atau firewall, Anda harus mengidentifikasi titik akhir untuk indeks paket dan repositori tempat perangkat inti Anda terhubung untuk menginstal.

Pertimbangkan hal berikut ketika Anda mengidentifikasi titik akhir yang diperlukan untuk skrip penginstalan komponen ini:

- Titik akhir bergantung pada platform perangkat inti. Misalnya, perangkat inti yang menjalankan Ubuntu menggunakan `apt` bukan `yum` atau`brew`. Selain itu, perangkat yang menggunakan indeks paket yang sama mungkin memiliki daftar sumber yang berbeda, sehingga mereka mungkin mengambil paket dari repositori yang berbeda.
- Titik akhir mungkin berbeda antara beberapa perangkat yang menggunakan indeks paket yang sama, karena setiap perangkat memiliki daftar sumbernya sendiri yang menentukan tempat untuk mengambil paket.
- Titik akhir mungkin berubah seiring waktu. Setiap indeks paket menyediakan URL repositori tempat Anda mengunduh paket, dan pemilik paket dapat mengubah URL apa yang disediakan indeks paket.

Untuk informasi selengkapnya tentang dependensi yang dipasang komponen ini, dan cara menonaktifkan skrip penginstal, lihat parameter konfigurasi. UseInstaller

Untuk informasi selengkapnya tentang titik akhir dan port yang diperlukan untuk operasi dasar, lihatIzinkan lalu lintas perangkat melalui proxy atau firewall.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

1.6.11 - 1.6.16

Tabel berikut mencantumkan dependensi untuk versi 1.6.11 hingga 1.6.16 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <3.0.0	Lunak

1.6.10

Tabel berikut mencantumkan dependensi untuk versi 1.6.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak

1.6.9

Tabel berikut mencantumkan dependensi untuk versi 1.6.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak

1.6.8

Tabel berikut mencantumkan dependensi untuk versi 1.6.8 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak

1.6.6 and 1.6.7

Tabel berikut mencantumkan dependensi untuk versi 1.6.6 dan 1.6.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak

1.6.4 and 1.6.5

Tabel berikut mencantumkan dependensi untuk versi 1.6.4 dan 1.6.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak

1.6.3

Tabel berikut mencantumkan dependensi untuk versi 1.6.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak

1.6.2

Tabel berikut mencantumkan dependensi untuk versi 1.6.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak

1.6.1

Tabel berikut mencantumkan dependensi untuk versi 1.6.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak

1.3.x

Tabel berikut mencantumkan dependensi untuk versi 1.3.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	~2.0.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`MLRootPath`

(Opsional) Jalur folder pada perangkat inti Linux tempat komponen inferensi membaca gambar dan menulis hasil inferensi. Anda dapat mengubah nilai ini ke lokasi mana pun di perangkat Anda yang padanya pengguna menjalankan komponen ini memiliki akses baca/tulis.

Default: `/greengrass/v2/work/variant.DLR/greengrass_ml`

`WindowsMLRootPath`

Fitur ini tersedia di v1.6.6 dan yang lebih baru dari komponen ini.

(Opsional) Jalur folder pada perangkat inti Windows tempat komponen inferensi membaca gambar dan menulis hasil inferensi. Anda dapat mengubah nilai ini ke lokasi mana pun di perangkat Anda yang padanya pengguna menjalankan komponen ini memiliki akses baca/tulis.

Default: *C:\greengrass\v2*\\work\\variant.DLR\\greengrass_ml

`UseInstaller`

(Opsional) Nilai string yang menentukan apakah akan menggunakan skrip installer dalam komponen ini untuk menginstal DLR dan dependensinya. Nilai yang didukung adalah `true` dan `false`.

Tetapkan nilai ini `false` jika Anda ingin menggunakan skrip kustom untuk instalasi DLR, atau jika Anda ingin menyertakan dependensi runtime dalam image Linux yang sudah dibuat sebelumnya. Untuk menggunakan komponen ini dengan komponen inferensi DLR yang AWS disediakan, instal pustaka berikut, termasuk dependensi apa pun, dan buat tersedia bagi pengguna sistem, seperti`gc_user`, yang menjalankan komponen ML.

- Python 3.7 atau yang lebih baru, termasuk `pip` untuk versi Python Anda.
- Runtime Pembelajaran Mendalam v1.6.0
- NumPy.
- OpenCV-Python.
- AWS IoT Device SDK v2 untuk Python.
- AWS Python Runtime Umum (CRT).
- Picamera (hanya untuk perangkat Raspberry Pi).
- `awscammodul` (untuk AWS DeepLens perangkat).
- LibGL (untuk perangkat Linux)

Default: `true`

Penggunaan

Gunakan komponen ini dengan parameter `UseInstaller` konfigurasi yang disetel `true` untuk menginstal DLR dan dependensinya di perangkat Anda. Komponen menyiapkan lingkungan virtual di perangkat Anda yang menyertakan OpenCV dan NumPy pustaka yang diperlukan untuk DLR.

> **Note**
>
> Skrip penginstal dalam komponen ini juga menginstal versi terbaru dari pustaka sistem tambahan yang diperlukan untuk mengonfigurasi lingkungan virtual pada perangkat Anda dan menggunakan kerangka kerja pembelajaran mesin yang diinstal. Hal ini dapat meningkatkan pustaka sistem yang ada di perangkat Anda. Tinjau tabel berikut untuk daftar pustaka yang menginstal komponen ini untuk setiap sistem operasi yang didukung. Jika Anda ingin menyesuaikan proses instalasi ini, atur parameter `UseInstaller` konfigurasi ke`false`, dan kembangkan skrip penginstal Anda sendiri.

Platform	Pustaka terpasang pada sistem perangkat	Pustaka terpasang di lingkungan virtual
Armv7l	`build-essential` , `cmake`, `ca-certificates` , `git`	`setuptools` , `wheel`
Amazon Linux 2	`mesa-libGL`	Tidak ada
Ubuntu	`wget`	Tidak ada

Ketika Anda men-deploy komponen inferensi Anda, komponen waktu aktif ini pertama-tama akan memverifikasi apakah perangkat Anda sudah memiliki DLR dan dependensinya sudah diinstal, dan jika tidak, ia akan menginstalnya untuk Anda.

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/variant.DLR.log
```

Windows

```
C:\greengrass\v2\logs\variant.DLR.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\variant.DLR.log -Tail 10 -Wait
  ```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
1.6.16	Versi diperbarui untuk Greengrass nucleus versi 2.12.5.
1.6.12	Perbaikan bug dan peningkatan • Memperbaiki skrip instalasi untuk pengguna OS Windows.
1.6.11	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
1.6.10	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
1.6.9	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
1.6.8	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
1.6.7	Perbaikan bug dan peningkatan • Memperbarui skrip `UseInstaller` instalasi untuk menginstal LibGL, yang tidak tersedia secara default pada platform Linux tertentu.

Versi	Perubahan
	• Memperbarui skrip `UseInstaller` instalasi untuk selalu menggunakan Python 3.9 di lingkungan virtual komponen ini. Perubahan ini membantu memastikan kompatibilitas dengan pustaka lain.
1.6.6	Fitur baru • Menambahkan dukungan untuk perangkat inti yang menjalankan Windows. • Menambahkan parameter `WindowsMLRootPath` konfigurasi baru yang dapat Anda gunakan untuk mengonfigurasi folder hasil inferensi pada perangkat inti Windows.
1.6.5	Fitur baru • Menambahkan parameter `UseInstaller` konfigurasi baru yang dapat Anda gunakan untuk menonaktifkan skrip instalasi dalam komponen ini.
1.6.4	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
1.6.3	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
1.6.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
1.6.1	Fitur baru • Instal Deep Learning Runtime v1.6.0 dan dependensinya. • Tambahkan dukungan untuk menginstal DLR pada platform Armv8 (AArch64). Hal ini akan memperluas dukungan machine learning untuk perangkat inti Greengrass yang menjalankan NVIDIA Jetson, seperti Jetson Nano. Perbaikan bug dan peningkatan • Instal AWS IoT Device SDK di lingkungan virtual untuk membaca konfigurasi komponen dan menerapkan perubahan konfigurasi. • Peningkatan dan perbaikan kecil tambahan.
1.3.2	Versi awal. Menginstal DLR v1.3.0.

TensorFlow Klasifikasi gambar ringan

Komponen klasifikasi gambar TensorFlow Lite (`aws.greengrass.TensorFlowLiteImageClassification`) berisi kode inferensi sampel untuk melakukan inferensi klasifikasi gambar menggunakan runtime TensorFlow Lite dan model terkuantisasi 1.0 yang telah dilatih sebelumnya MobileNet . Komponen ini menggunakan varian TensorFlow Toko model klasifikasi gambar Lite dan TensorFlow Runtime ringan komponen sebagai dependensi untuk mengunduh runtime TensorFlow Lite dan model sampel.

Untuk menggunakan komponen inferensi ini dengan model TensorFlow Lite yang terlatih khusus, buat versi kustom komponen penyimpanan model dependen. Untuk menggunakan kode inferensi kustom Anda sendiri, Anda dapat menggunakan resep komponen ini sebagai templat untuk membuat komponen inferensi kustom.

Topik

- Versi
- Jenis
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- File log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.1.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat.
- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```
- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

    ```
    pip3 install --upgrade numpy
    ```
  - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

    Untuk mengaktifkan tumpukan kamera lama

    1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

       ```
       sudo raspi-config
       ```
    2. Pilih Opsi Antarmuka.
    3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.

4. Reboot Raspberry Pi.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.11 and 2.1.12

Tabel berikut mencantumkan dependensi untuk versi 2.1.11 dan 2.1.12 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.10

Tabel berikut mencantumkan dependensi untuk versi 2.1.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.9

Tabel berikut mencantumkan dependensi untuk versi 2.1.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`accessControl`

(Opsional) Objek yang berisi kebijakan otorisasi yang memungkinkan komponen untuk mempublikasikan pesan ke topik pemberitahuan default.

Default:

```
{
   "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.TensorFlowLiteImageClassification:mqttproxy:1": {
         "policyDescription": "Allows access to publish via topic ml/tflite/image-
classification.",
         "operations": [
            "aws.greengrass#PublishToIoTCore"
         ],
         "resources": [
            "ml/tflite/image-classification"
         ]
      }
   }
}
```

`PublishResultsOnTopic`

(Opsional) Topik di mana Anda ingin mempublikasikan hasil inferensi. Jika Anda mengubah nilai ini, maka Anda juga harus mengubah nilai `resources` di parameter `accessControl` agar cocok dengan nama topik kustom Anda.

Default: `ml/tflite/image-classification`

`Accelerator`

Akselerator yang ingin Anda gunakan. Nilai yang didukung adalah `cpu` dan `gpu`.

Model sampel dalam komponen model dependen hanya mendukung akselerasi CPU. Untuk menggunakan akselerasi GPU dengan model kustom yang berbeda, buat komponen model kustom untuk menimpa komponen model publik.

Default: `cpu`

`ImageDirectory`

(Opsional) Jalur folder pada perangkat di mana komponen inferensi membaca gambar. Anda dapat mengubah nilai ini ke lokasi mana pun di perangkat Anda yang memiliki akses baca/tulis.

Default: */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/image_classification/sample_images/`

> **Note**
>
> Jika Anda menetapkan nilai `UseCamera` ke `true`, maka parameter konfigurasi ini diabaikan.

`ImageName`

(Opsional) Nama gambar yang digunakan oleh komponen inferensi sebagai masukan untuk membuat prediksi. Komponen mencari gambar dalam folder yang ditentukan dalam `ImageDirectory`. Secara default, komponen menggunakan gambar sampel di direktori gambar default. AWS IoT Greengrass mendukung format gambar berikut:`jpeg`,`jpg`,`png`, dannpy.

Default: `cat.jpeg`

> **Note**
>
> Jika Anda menetapkan nilai `UseCamera` ke `true`, maka parameter konfigurasi ini diabaikan.

`InferenceInterval`

(Opsional) Waktu dalam detik antara setiap prediksi yang dibuat oleh kode inferensi. Kode inferensi sampel berjalan tanpa batas waktu dan mengulangi prediksinya pada interval waktu yang ditentukan. Misalnya, Anda dapat mengubahnya menjadi interval yang lebih pendek jika ingin menggunakan gambar yang diambil oleh kamera untuk prediksi real-time.

Default: `3600`

`ModelResourceKey`

(Opsional) Model yang digunakan dalam komponen model publik dependen. Ubah parameter ini hanya jika Anda menimpa komponen model publik dengan komponen kustom.

Default:

```
{
    "model": "TensorFlowLite-Mobilenet"
}
```

`UseCamera`

(Opsional) Nilai string yang menentukan apakah akan menggunakan gambar dari kamera yang terhubung ke perangkat inti Greengrass. Nilai yang didukung adalah `true` dan `false`.

Ketika Anda menetapkan nilai ini ke `true`, kode kesimpulan sampel akan mengakses kamera pada perangkat Anda dan menjalankan kesimpulan secara lokal pada gambar yang ditangkap. Nilai `ImageName` dan parameter `ImageDirectory` diabaikan. Pastikan bahwa pengguna yang menjalankan komponen ini memiliki akses baca/tulis ke lokasi di mana kamera menyimpan gambar yang diambil.

Default: `false`

> **Note**
>
> Ketika Anda melihat resep komponen ini, parameter konfigurasi `UseCamera` tidak muncul dalam konfigurasi default. Namun, Anda dapat mengubah nilai parameter ini dalam pembaruan gabungan konfigurasi saat Anda men-deploy komponen.
> Ketika Anda mengatur `UseCamera` ke `true`, Anda juga harus membuat symlink untuk mengaktifkan komponen inferensi untuk mengakses kamera Anda dari lingkungan virtual yang dibuat oleh komponen waktu aktif. Untuk informasi lebih lanjut tentang penggunaan kamera dengan komponen inferensi sampel, lihat Perbarui konfigurasi komponen.

File log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.greengrass.TensorFlowLiteImageClassification.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.TensorFlowLiteImageClassification.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs
  \aws.greengrass.TensorFlowLiteImageClassification.log -Tail 10 -Wait
  ```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.12	Versi diperbarui untuk Greengrass nucleus 2.12.5 rilis.
2.1.11	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.10	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.9	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.5	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.1.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.1.0	Versi awal.

TensorFlow Deteksi objek Lite

Komponen deteksi objek TensorFlow Lite (`aws.greengrass.TensorFlowLiteObjectDetection`) berisi kode inferensi sampel untuk melakukan inferensi deteksi objek menggunakan TensorFlow Lite dan model Single Shot Detection (SSD) MobileNet 1.0 yang telah dilatih sebelumnya. Komponen ini menggunakan varian TensorFlow Toko model deteksi objek Lite dan TensorFlow Runtime ringan komponen sebagai dependensi untuk mengunduh TensorFlow Lite dan model sampel.

Untuk menggunakan komponen inferensi ini dengan model TensorFlow Lite yang terlatih khusus, Anda dapat membuat versi kustom komponen penyimpanan model dependen. Untuk menggunakan kode inferensi kustom Anda sendiri, gunakan resep komponen ini sebagai templat untuk membuat komponen inferensi kustom.

Topik

- Versi
- Jenis
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- Berkas log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.1.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat.
- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```
- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

    ```
    pip3 install --upgrade numpy
    ```
  - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

    Untuk mengaktifkan tumpukan kamera lama

    1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

       ```
       sudo raspi-config
       ```
    2. Pilih Opsi Antarmuka.
    3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.
    4. Reboot Raspberry Pi.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi

yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.11 and 2.1.12

Tabel berikut mencantumkan dependensi untuk versi 2.1.11 dan 2.1.12 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.10

Tabel berikut mencantumkan dependensi untuk versi 2.1.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.9

Tabel berikut mencantumkan dependensi untuk versi 2.1.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak

Dependensi	Versi yang kompatibel	Jenis dependensi
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak
TensorFlow Toko model klasifikasi gambar Lite	>=2.1.0 <2.2.0	Keras
TensorFlow Lite	>=2.5.0 <2.6.0	Keras

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`accessControl`

(Opsional) Objek yang berisi kebijakan otorisasi yang memungkinkan komponen untuk mempublikasikan pesan ke topik pemberitahuan default.

Default:

```
{
   "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.TensorFlowLiteObjectDetection:mqttproxy:1": {
         "policyDescription": "Allows access to publish via topic ml/tflite/object-
detection.",
         "operations": [
            "aws.greengrass#PublishToIoTCore"
         ],
         "resources": [
            "ml/tflite/object-detection"
         ]
      }
   }
}
```

`PublishResultsOnTopic`

(Opsional) Topik di mana Anda ingin mempublikasikan hasil inferensi. Jika Anda mengubah nilai ini, maka Anda juga harus mengubah nilai `resources` di parameter `accessControl` agar cocok dengan nama topik kustom Anda.

Default: `ml/tflite/object-detection`

`Accelerator`

Akselerator yang ingin Anda gunakan. Nilai yang didukung adalah `cpu` dan `gpu`.

Model sampel dalam komponen model dependen hanya mendukung akselerasi CPU. Untuk menggunakan akselerasi GPU dengan model kustom yang berbeda, buat komponen model kustom untuk menimpa komponen model publik.

Default: `cpu`

`ImageDirectory`

(Opsional) Jalur folder pada perangkat di mana komponen inferensi membaca gambar. Anda dapat mengubah nilai ini ke lokasi mana pun di perangkat Anda yang memiliki akses baca/tulis.

Default: *`/greengrass/v2`*`/packages/artifacts-unarchived/`*`component-name`*`/object_detection/sample_images/`

> **Note**
>
> Jika Anda menetapkan nilai `UseCamera` ke `true`, maka parameter konfigurasi ini diabaikan.

`ImageName`

(Opsional) Nama gambar yang digunakan oleh komponen inferensi sebagai masukan untuk membuat prediksi. Komponen mencari gambar dalam folder yang ditentukan dalam `ImageDirectory`. Secara default, komponen menggunakan gambar sampel di direktori gambar default. AWS IoT Greengrass mendukung format gambar berikut:`jpeg`,`jpg`,`png`, dan`npy`.

Default: `objects.jpg`

> **Note**
>
> Jika Anda menetapkan nilai `UseCamera` ke `true`, maka parameter konfigurasi ini diabaikan.

`InferenceInterval`

(Opsional) Waktu dalam detik antara setiap prediksi yang dibuat oleh kode inferensi. Kode inferensi sampel berjalan tanpa batas waktu dan mengulangi prediksinya pada interval waktu yang ditentukan. Misalnya, Anda dapat mengubahnya menjadi interval yang lebih pendek jika ingin menggunakan gambar yang diambil oleh kamera untuk prediksi real-time.

Default: `3600`

`ModelResourceKey`

(Opsional) Model yang digunakan dalam komponen model publik dependen. Ubah parameter ini hanya jika Anda menimpa komponen model publik dengan komponen kustom.

Default:

```
{
    "model": "TensorFlowLite-SSD"
}
```

`UseCamera`

(Opsional) Nilai string yang menentukan apakah akan menggunakan gambar dari kamera yang terhubung ke perangkat inti Greengrass. Nilai yang didukung adalah `true` dan `false`.

Ketika Anda menetapkan nilai ini ke `true`, kode kesimpulan sampel akan mengakses kamera pada perangkat Anda dan menjalankan kesimpulan secara lokal pada gambar yang ditangkap. Nilai `ImageName` dan parameter `ImageDirectory` diabaikan. Pastikan bahwa pengguna yang menjalankan komponen ini memiliki akses baca/tulis ke lokasi di mana kamera menyimpan gambar yang diambil.

Default: `false`

> **Note**
>
> Ketika Anda melihat resep komponen ini, parameter konfigurasi `UseCamera` tidak muncul dalam konfigurasi default. Namun, Anda dapat mengubah nilai parameter ini dalam pembaruan gabungan konfigurasi saat Anda men-deploy komponen.
> Ketika Anda mengatur `UseCamera` ke `true`, Anda juga harus membuat symlink untuk mengaktifkan komponen inferensi untuk mengakses kamera Anda dari lingkungan virtual

yang dibuat oleh komponen waktu aktif. Untuk informasi lebih lanjut tentang penggunaan kamera dengan komponen inferensi sampel, lihat Perbarui konfigurasi komponen.

> **Note**
>
> Ketika Anda melihat resep komponen ini, parameter konfigurasi `UseCamera` tidak muncul dalam konfigurasi default. Namun, Anda dapat mengubah nilai parameter ini dalam pembaruan gabungan konfigurasi saat Anda men-deploy komponen.
> Ketika Anda mengatur `UseCamera` ke `true`, Anda juga harus membuat symlink untuk mengaktifkan komponen inferensi untuk mengakses kamera Anda dari lingkungan virtual yang dibuat oleh komponen waktu aktif. Untuk informasi lebih lanjut tentang penggunaan kamera dengan komponen inferensi sampel, lihat Perbarui konfigurasi komponen.

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.greengrass.TensorFlowLiteObjectDetection.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.TensorFlowLiteObjectDetection.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs
\aws.greengrass.TensorFlowLiteObjectDetection.log -Tail 10 -Wait
```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.12	Versi diperbarui untuk Greengrass nucleus 2.12.5 rilis.
2.1.11	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.10	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.9	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.5	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.1.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.1	Perbaikan bug dan peningkatan • Memperbaiki masalah penskalaan gambar yang mengakibatkan kotak pembatas tidak akurat dalam hasil inferensi deteksi objek TensorFlow Lite sampel.
2.1.0	Versi awal.

TensorFlow Toko model klasifikasi gambar Lite

Toko model klasifikasi gambar TensorFlow Lite (`variant.TensorFlowLite.ImageClassification.ModelStore`) adalah komponen model pembelajaran mesin yang berisi model MobileNet v1 yang telah dilatih sebelumnya sebagai artefak Greengrass. Model sampel yang digunakan dalam komponen ini diambil dari TensorFlowHub dan diimplementasikan menggunakan TensorFlow Lite.

Komponen inferensi TensorFlow Klasifikasi gambar ringan menggunakan komponen ini sebagai dependensi untuk sumber model. Untuk menggunakan model TensorFlow Lite yang terlatih khusus, buat versi kustom komponen model ini, dan sertakan model kustom Anda sebagai artefak komponen. Anda dapat menggunakan resep komponen ini sebagai templat untuk membuat komponen model kustom.

Topik

- Versi
- Jenis
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- Berkas log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.1.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat.
- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```
- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

    ```
    pip3 install --upgrade numpy
    ```
  - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

    Untuk mengaktifkan tumpukan kamera lama

    1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

       ```
       sudo raspi-config
       ```
    2. Pilih Opsi Antarmuka.
    3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.

4. Reboot Raspberry Pi.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.11 and 2.1.12

Tabel berikut mencantumkan dependensi untuk versi 2.1.11 dan 2.1.12 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak

2.1.10

Tabel berikut mencantumkan dependensi untuk versi 2.1.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak

2.1.9

Tabel berikut mencantumkan dependensi untuk versi 2.1.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak

2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak

2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

Berkas log lokal

Komponen ini tidak mengeluarkan log.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.12	Versi diperbarui untuk Greengrass nucleus 2.12.5 rilis.
2.1.11	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.10	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.9	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.5	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.1.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.1.0	Versi awal.

TensorFlow Toko model deteksi objek Lite

TensorFlow Lite object detection model store (`variant.TensorFlowLite.ObjectDetection.ModelStore`) adalah komponen model pembelajaran mesin yang berisi model Single Shot Detection (SSD) MobileNet pra-terlatih sebagai artefak Greengrass. Model sampel yang digunakan dalam komponen ini diambil dari TensorFlow Hub dan diimplementasikan menggunakan TensorFlow Lite.

Komponen inferensi deteksi objek TensorFlow Lite menggunakan komponen ini sebagai dependensi untuk sumber model. Untuk menggunakan model TensorFlow Lite yang terlatih khusus, buat versi kustom komponen model ini, dan sertakan model kustom Anda sebagai artefak komponen. Anda dapat menggunakan resep komponen ini sebagai templat untuk membuat komponen model kustom.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.1.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat.
- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```
- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

    ```
    pip3 install --upgrade numpy
    ```
  - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

    Untuk mengaktifkan tumpukan kamera lama

    1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

       ```
       sudo raspi-config
       ```
    2. Pilih Opsi Antarmuka.
    3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.

4. Reboot Raspberry Pi.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.11 and 2.1.12

Tabel berikut mencantumkan dependensi untuk versi 2.1.11 dan 2.1.12 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak

2.1.10

Tabel berikut mencantumkan dependensi untuk versi 2.1.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak

2.1.9

Tabel berikut mencantumkan dependensi untuk versi 2.1.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak

2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak

2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

Berkas log lokal

Komponen ini tidak mengeluarkan log.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.12	Versi diperbarui untuk Greengrass nucleus 2.12.5 rilis.
2.1.11	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.10	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.9	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.5	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.1.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.1.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.1.0	Versi awal.

TensorFlow Runtime ringan

Komponen runtime TensorFlow Lite (`variant.TensorFlowLite`) berisi skrip yang menginstal TensorFlow Lite versi 2.5.0 dan dependensinya di lingkungan virtual di perangkat Anda. Klasifikasi gambar TensorFlow TensorFlow Lite dan komponen deteksi objek Lite menggunakan komponen runtime ini sebagai dependensi untuk menginstal TensorFlow Lite.

> **Note**
>
> TensorFlow Komponen runtime Lite v2.5.6 dan yang lebih baru menginstal ulang instalasi yang ada dari runtime Lite dan dependensinya. TensorFlow Instalasi ulang ini membantu memastikan bahwa perangkat inti menjalankan versi TensorFlow Lite yang kompatibel dan dependensinya.

Untuk menggunakan runtime yang berbeda, Anda dapat menggunakan resep komponen ini sebagai template untuk membuat komponen pembelajaran mesin kustom.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.5.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat.
- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

    ```
    pip3 install --upgrade numpy
    ```

  - Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

Untuk mengaktifkan tumpukan kamera lama

1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

   ```
   sudo raspi-config
   ```

2. Pilih Opsi Antarmuka.
3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.
4. Reboot Raspberry Pi.

Titik akhir dan port

Secara default, komponen ini menggunakan skrip installer untuk menginstal paket menggunakan`apt`,, `yum``brew`, dan `pip` perintah, tergantung pada platform apa yang digunakan perangkat inti. Komponen ini harus dapat melakukan permintaan keluar ke berbagai indeks paket dan repositori untuk menjalankan skrip installer. Untuk mengizinkan lalu lintas keluar komponen ini melalui proxy atau firewall, Anda harus mengidentifikasi titik akhir untuk indeks paket dan repositori tempat perangkat inti Anda terhubung untuk menginstal.

Pertimbangkan hal berikut ketika Anda mengidentifikasi titik akhir yang diperlukan untuk skrip penginstalan komponen ini:

- Titik akhir bergantung pada platform perangkat inti. Misalnya, perangkat inti yang menjalankan Ubuntu menggunakan `apt` bukan `yum` atau`brew`. Selain itu, perangkat yang menggunakan indeks paket yang sama mungkin memiliki daftar sumber yang berbeda, sehingga mereka mungkin mengambil paket dari repositori yang berbeda.
- Titik akhir mungkin berbeda antara beberapa perangkat yang menggunakan indeks paket yang sama, karena setiap perangkat memiliki daftar sumbernya sendiri yang menentukan tempat untuk mengambil paket.
- Titik akhir mungkin berubah seiring waktu. Setiap indeks paket menyediakan URL repositori tempat Anda mengunduh paket, dan pemilik paket dapat mengubah URL apa yang disediakan indeks paket.

Untuk informasi selengkapnya tentang dependensi yang dipasang komponen ini, dan cara menonaktifkan skrip penginstal, lihat parameter konfigurasi. UseInstaller

Untuk informasi selengkapnya tentang titik akhir dan port yang diperlukan untuk operasi dasar, lihatIzinkan lalu lintas perangkat melalui proxy atau firewall.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.5.14 and 2.5.15

Tabel berikut mencantumkan dependensi untuk versi 2.5.14 dan 2.5.15 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak

2.5.13

Tabel berikut mencantumkan dependensi untuk versi 2.5.13 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak

2.5.12

Tabel berikut mencantumkan dependensi untuk versi 2.5.12 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak

2.5.11

Tabel berikut mencantumkan dependensi untuk versi 2.5.11 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak

2.5.10

Tabel berikut mencantumkan dependensi untuk versi 2.5.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak

2.5.9

Tabel berikut mencantumkan dependensi untuk versi 2.5.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak

2.5.8

Tabel berikut mencantumkan dependensi untuk versi 2.5.8 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak

2.5.5 - 2.5.7

Tabel berikut mencantumkan dependensi untuk versi 2.5.5 hingga 2.5.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak

2.5.3 and 2.5.4

Tabel berikut mencantumkan dependensi untuk versi 2.5.3 dan 2.5.4 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak

2.5.2

Tabel berikut mencantumkan dependensi untuk versi 2.5.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak

2.5.1

Tabel berikut mencantumkan dependensi untuk versi 2.5.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak

2.5.0

Tabel berikut mencantumkan dependensi untuk versi 2.5.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`MLRootPath`

(Opsional) Jalur folder pada perangkat inti Linux tempat komponen inferensi membaca gambar dan menulis hasil inferensi. Anda dapat mengubah nilai ini ke lokasi mana pun di perangkat Anda yang padanya pengguna menjalankan komponen ini memiliki akses baca/tulis.

Default: */greengrass/v2*`/work/variant.TensorFlowLite/greengrass_ml`

`WindowsMLRootPath`

Fitur ini tersedia di v1.6.6 dan yang lebih baru dari komponen ini.

(Opsional) Jalur folder pada perangkat inti Windows tempat komponen inferensi membaca gambar dan menulis hasil inferensi. Anda dapat mengubah nilai ini ke lokasi mana pun di perangkat Anda yang padanya pengguna menjalankan komponen ini memiliki akses baca/tulis.

Default: *C:\greengrass\v2*`\\work\\variant.DLR\\greengrass_ml`

`UseInstaller`

(Opsional) Nilai string yang menentukan apakah akan menggunakan skrip installer dalam komponen ini untuk menginstal TensorFlow Lite dan dependensinya. Nilai yang didukung adalah `true` dan `false`.

Tetapkan nilai ini `false` jika Anda ingin menggunakan skrip kustom untuk instalasi TensorFlow Lite, atau jika Anda ingin menyertakan dependensi runtime dalam image Linux yang sudah dibuat sebelumnya. Untuk menggunakan komponen ini dengan komponen inferensi TensorFlow Lite AWS-provided, instal pustaka berikut, termasuk dependensi apa pun, dan sediakan komponen tersebut bagi pengguna sistem, seperti`ggc_user`, yang menjalankan komponen ML.

- Python 3.8 atau yang lebih baru, termasuk `pip` untuk versi Python
- TensorFlow Lite v2.5.0
- NumPy
- OpenCV-Python

- AWS IoT Device SDK v2 untuk Python
- AWS Python Runtime Umum (CRT)
- Picamera (untuk perangkat Raspberry Pi)
- `awscammodul` (untuk AWS DeepLens perangkat)
- LibGL (untuk perangkat Linux)

Default: `true`

Penggunaan

Gunakan komponen ini dengan parameter `UseInstaller` konfigurasi yang disetel `true` untuk menginstal TensorFlow Lite dan dependensinya di perangkat Anda. Komponen menyiapkan lingkungan virtual di perangkat Anda yang menyertakan OpenCV dan NumPy pustaka yang diperlukan untuk Lite. TensorFlow

> **Note**
>
> Skrip penginstal dalam komponen ini juga menginstal versi terbaru dari pustaka sistem tambahan yang diperlukan untuk mengonfigurasi lingkungan virtual pada perangkat Anda dan menggunakan kerangka kerja pembelajaran mesin yang diinstal. Hal ini dapat meningkatkan pustaka sistem yang ada di perangkat Anda. Tinjau tabel berikut untuk daftar pustaka yang menginstal komponen ini untuk setiap sistem operasi yang didukung. Jika Anda ingin menyesuaikan proses instalasi ini, atur parameter `UseInstaller` konfigurasi `kefalse`, dan kembangkan skrip penginstal Anda sendiri.

Platform	Pustaka terpasang pada sistem perangkat	Pustaka terpasang di lingkungan virtual
Armv7l	`build-essential` , `cmake`, `ca-certificates` , `git`	`setuptools` , `wheel`
Amazon Linux 2	`mesa-libGL`	Tidak ada
Ubuntu	`wget`	Tidak ada

Saat Anda menerapkan komponen inferensi, komponen runtime ini terlebih dahulu memverifikasi apakah perangkat Anda sudah menginstal TensorFlow Lite dan dependensinya. Jika tidak, maka komponen waktu aktif akan menginstalnya untuk Anda.

File log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/variant.TensorFlowLite.log
```

Windows

```
C:\greengrass\v2\logs\variant.TensorFlowLite.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\variant.TensorFlowLite.log -Tail 10 -Wait
  ```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.5.15	Versi diperbarui untuk Greengrass nucleus 2.12.5 rilis.

Versi	Perubahan
2.5.14	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.5.13	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.5.12	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.5.11	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.5.10	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.5.9	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.5.8	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.5.7	Perbaikan bug dan peningkatan • Memperbarui skrip `UseInstaller` instalasi untuk menginstal LibGL, yang tidak tersedia secara default pada platform Linux tertentu. • Memperbarui skrip `UseInstaller` instalasi untuk selalu menggunakan Python 3.9 di lingkungan virtual komponen ini. Perubahan ini membantu memastikan kompatibilitas dengan pustaka lain.
2.5.6	Perbaikan bug dan peningkatan • Memperbarui komponen ini untuk menginstal patch terbaru TensorFlow Lite 2.5.0 (`tflite-runtime-2.5.0.post1`), sehingga Anda dapat menggunakan komponen ini dengan Python 3.9. Jika komponen ini gagal menginstal tambalan itu, ia akan menginstal `tflite-runtime-2.5.0` sebagai gantinya. • Memperbarui komponen ini untuk menginstal ulang instalasi TensorFlow Lite yang ada dan dependensinya. Perubahan ini membantu memastikan bahwa perangkat inti menjalankan versi TensorFlow Lite yang kompatibel dan dependensinya.

Versi	Perubahan
2.5.5	Fitur baru • Menambahkan dukungan untuk perangkat inti yang menjalankan Windows. • Menambahkan parameter `WindowsMLRootPath` konfigurasi baru yang dapat Anda gunakan untuk mengonfigurasi folder hasil inferensi pada perangkat inti Windows.
2.5.4	Fitur baru • Menambahkan parameter `UseInstaller` konfigurasi baru yang memungkinkan Anda menonaktifkan skrip instalasi dalam komponen ini.
2.5.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.5.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.5.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.5.0	Versi awal.

Adaptor protokol Modbus-RTU

Komponen adaptor protokol Modbus-RTU (`aws.greengrass.Modbus`) mengumpulkan informasi dari perangkat Modbus RTU lokal.

Untuk meminta informasi dari perangkat Modbus RTU lokal dengan komponen ini, publikasikan pesan ke topik di mana komponen ini berlangganan. Dalam pesan, tentukan permintaan Modbus RTU yang akan dikirim ke perangkat. Kemudian, komponen ini akan menerbitkan respon yang berisi hasil permintaan Modbus RTU.

> **Note**
>
> Komponen ini menyediakan fungsionalitas yang mirip dengan konektor adaptor protokol Modbus RTU di AWS IoT Greengrass V1. Untuk informasi selengkapnya, lihat konektor adaptor protokol RTU Modbus di Panduan Developer V1 AWS IoT Greengrass .

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen Lambda () `aws.greengrass.lambda`. Inti Greengrass menjalankan fungsi Lambda komponen ini menggunakan komponen peluncur Lambda.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Anda harus memenuhi persyaratan untuk menjalankan fungsi Lambda. Jika Anda ingin perangkat inti untuk menjalankan fungsi Lambda kontainer, perangkat harus memenuhi persyaratan untuk melakukannya. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.
- Python versi 3.7 diinstal pada perangkat inti dan ditambahkan ke variabel lingkungan PATH.
- Koneksi fisik antara perangkat AWS IoT Greengrass inti dan perangkat Modbus. Perangkat inti harus terhubung secara fisik ke jaringan Modbus RTU melalui port serial, seperti port USB.
- Untuk menerima data keluaran dari komponen ini, Anda harus menggabungkan pemutakhiran konfigurasi berikut untuk komponen router langganan lama (`aws.greengrass.LegacySubscriptionRouter`) saat menerapkan komponen ini. Konfigurasi ini menentukan topik di mana komponen ini menerbitkan tanggapan.

  Legacy subscription router v2.1.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-modbus": {
        "id": "aws-greengrass-modbus",
        "source": "component:aws.greengrass.Modbus",
        "subject": "modbus/adapter/response",
        "target": "cloud"
      }
    }
  }
  ```

  Legacy subscription router v2.0.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-modbus": {
        "id": "aws-greengrass-modbus",
        "source": "arn:aws:lambda:region:aws:function:aws-greengrass-
  modbus:version",
        "subject": "modbus/adapter/response",
        "target": "cloud"
      }
    }
  }
  ```

  - Ganti *wilayah* dengan Wilayah AWS yang Anda gunakan.

- Ganti *versi* dengan versi fungsi Lambda yang komponen ini jalankan. Untuk menemukan versi fungsi Lambda, Anda harus melihat resep untuk versi komponen ini yang ingin Anda deploy. Buka halaman detail komponen ini di konsol AWS IoT Greengrass tersebut, dan cari pasangan nilai kunci fungsi Lambda. Pasangan kunci-nilai ini berisi nama dan versi fungsi Lambda.

  > **Important**
  >
  > Anda harus memperbarui versi fungsi Lambda pada router langganan warisan setiap kali Anda men-deploy komponen ini. Hal ini memastikan bahwa Anda menggunakan versi fungsi Lambda yang benar untuk versi komponen yang Anda deploy.

  Untuk informasi selengkapnya, lihat Buat deployment.
- Adaptor protokol Modbus-RTU didukung untuk berjalan di VPC.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.4 and 2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 dan 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	^2.0.0	Keras

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.8 and 2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.8 dan 2.1.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.7

Tabel berikut mencantumkan dependensi untuk versi 2.0.7 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.6

Tabel berikut mencantumkan dependensi untuk versi 2.0.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.5

Tabel berikut mencantumkan dependensi untuk versi 2.0.5 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.4

Tabel berikut mencantumkan dependensi untuk versi 2.0.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	^2.0.0	Keras

2.0.3

Tabel berikut mencantumkan dependensi untuk versi 2.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Keras
Peluncur Lambda	>=1.0.0	Keras
Runtime Lambda	>=1.0.0	Lunak
Layanan pertukaran Token	>=1.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

> **Note**
>
> Konfigurasi default komponen ini meliputi parameter fungsi Lambda. Kami sarankan Anda mengedit hanya parameter berikut untuk mengonfigurasi komponen ini pada perangkat Anda.

v2.1.x

`lambdaParams`

Sebuah objek yang berisi parameter untuk fungsi Lambda komponen ini. Objek ini berisi informasi berikut:

`EnvironmentVariables`

Sebuah objek yang berisi parameter fungsi Lambda ini. Objek ini berisi informasi berikut:

`ModbusLocalPort`

Jalur absolut ke port serial Modbus fisik pada perangkat inti, seperti`/dev/ttyS2`.

Untuk menjalankan komponen ini dalam wadah, Anda harus mendefinisikan jalur ini sebagai perangkat sistem (in`containerParams.devices`) yang dapat diakses komponen. Komponen berjalan dalam kontainer secara default.

> **Note**
>
> Komponen ini harus memiliki akses baca/tulis ke perangkat.

`ModbusBaudRate`

(Opsional) Nilai string yang menentukan baud rate untuk komunikasi serial dengan perangkat Modbus TCP lokal.

Default: `9600`

`ModbusByteSize`

(Opsional) Nilai string yang menentukan ukuran byte dalam komunikasi serial dengan perangkat Modbus TCP lokal. Pilih5,6,7, atau 8 bit.

Default: `8`

`ModbusParity`

(Opsional) Mode paritas yang digunakan untuk memverifikasi integritas data dalam komunikasi serial dengan perangkat Modbus TCP lokal.

- `E`— Verifikasi integritas data dengan paritas yang merata.
- `0`— Verifikasi integritas data dengan paritas ganjil.
- `N`— Jangan memverifikasi integritas data.

Default: `N`

`ModbusStopBits`

(Opsional) Nilai string yang menentukan jumlah bit yang menunjukkan akhir byte dalam komunikasi serial dengan perangkat Modbus TCP lokal.

Default: 1

`containerMode`

(Opsional) Mode kontainerisasi untuk komponen ini. Pilih dari salah satu pilihan berikut:

- `GreengrassContainer`— Komponen berjalan di lingkungan runtime yang terisolasi di dalam AWS IoT Greengrass wadah.

  Jika Anda menentukan opsi ini, Anda harus menentukan perangkat sistem (in`containerParams.devices`) untuk memberikan akses wadah ke perangkat Modbus.
- `NoContainer` – Komponen tersebut tidak berjalan di lingkungan waktu aktif terisolasi.

Default: `GreengrassContainer`

`containerParams`

(Opsional) Sebuah objek yang berisi parameter kontainer untuk komponen ini. Komponen menggunakan parameter ini jika Anda menentukan `GreengrassContainer` untuk `containerMode`.

Objek ini berisi informasi berikut:

`memorySize`

(Opsional) Jumlah memori (dalam kilobyte) yang akan dialokasikan ke komponen.

Defaultnya 512 MB (525.312 KB).

`devices`

(Opsional) Sebuah objek yang menentukan perangkat sistem yang dapat diakses oleh komponen dalam kontainer.

> **Important**
>
> Untuk menjalankan komponen ini dalam sebuah kontainer, Anda harus menentukan perangkat sistem yang Anda konfigurasi di variabel lingkungan `ModbusLocalPort`.

Objek ini berisi informasi berikut:

`0` - Ini adalah indeks himpunan sebagai string.

Objek yang berisi informasi berikut:

`path`

Jalur ke perangkat sistem pada perangkat inti. Ini harus memiliki nilai yang sama dengan nilai yang Anda konfigurasi untuk `ModbusLocalPort`.

`permission`

(Opsional) Izin untuk mengakses perangkat sistem dari kontainer. Nilai ini harus `rw`, yang menentukan bahwa komponen memiliki akses baca/tulis ke perangkat sistem.

Default: `rw`

`addGroupOwner`

(Opsional) Apakah akan menambahkan grup sistem atau tidak yang akan menjalankan komponen sebagai pemilik perangkat sistem.

Default: `true`

`pubsubTopics`

(Opsional) Sebuah objek yang berisi topik di mana komponen berlangganan untuk menerima pesan. Anda dapat menentukan setiap topik dan apakah komponen berlangganan topik MQTT dari AWS IoT Core atau topik penerbitan/langganan lokal.

Objek ini berisi informasi berikut:

`0` - Ini adalah indeks himpunan sebagai string.

Objek yang berisi informasi berikut:

`type`

(Opsional) Jenis olahpesan publikasikan/berlangganan yang digunakan oleh komponen ini untuk berlangganan pesan. Pilih dari salah satu pilihan berikut:

- `PUB_SUB` — Berlangganan pesan publish/subscribe lokal. Jika Anda memilih opsi ini, topik tidak dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Pesan lokal publikasi/berlangganan.

- `IOT_CORE`— Berlangganan pesan AWS IoT Core MQTT. Jika Anda memilih opsi ini, topik dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Terbitkan/berlangganan pesan MQTT AWS IoT Core.

Default: `PUB_SUB`

`topic`

(Opsional) Topik yang menjadi langganan komponen untuk menerima pesan. Jika Anda menentukan `IotCore` untuk `type`, Anda dapat menggunakan wildcard MQTT (+ dan #) dalam topik ini.

Example Contoh: Pembaruan gabungan konfigurasi (mode kontainer)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "GreengrassContainer",
  "containerParams": {
    "devices": {
      "0": {
        "path": "/dev/ttyS2",
        "permission": "rw",
        "addGroupOwner": true
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (tidak ada mode kontainer)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "NoContainer"
```

```
}
```

v2.0.x

`lambdaParams`

Sebuah objek yang berisi parameter untuk fungsi Lambda komponen ini. Objek ini berisi informasi berikut:

`EnvironmentVariables`

Sebuah objek yang berisi parameter fungsi Lambda ini. Objek ini berisi informasi berikut:

`ModbusLocalPort`

Jalur absolut ke port serial Modbus fisik pada perangkat inti, seperti`/dev/ttyS2`.

Untuk menjalankan komponen ini dalam wadah, Anda harus mendefinisikan jalur ini sebagai perangkat sistem (in`containerParams.devices`) yang dapat diakses komponen. Komponen berjalan dalam kontainer secara default.

> **Note**
>
> Komponen ini harus memiliki akses baca/tulis ke perangkat.

`containerMode`

(Opsional) Mode kontainerisasi untuk komponen ini. Pilih dari salah satu pilihan berikut:

- `GreengrassContainer`— Komponen berjalan di lingkungan runtime yang terisolasi di dalam AWS IoT Greengrass wadah.

  Jika Anda menentukan opsi ini, Anda harus menentukan perangkat sistem (in`containerParams.devices`) untuk memberikan akses wadah ke perangkat Modbus.
- `NoContainer` – Komponen tersebut tidak berjalan di lingkungan waktu aktif terisolasi.

Default: `GreengrassContainer`

`containerParams`

(Opsional) Sebuah objek yang berisi parameter kontainer untuk komponen ini. Komponen menggunakan parameter ini jika Anda menentukan `GreengrassContainer` untuk `containerMode`.

Objek ini berisi informasi berikut:

`memorySize`

(Opsional) Jumlah memori (dalam kilobyte) yang akan dialokasikan ke komponen.

Defaultnya 512 MB (525.312 KB).

`devices`

(Opsional) Sebuah objek yang menentukan perangkat sistem yang dapat diakses oleh komponen dalam kontainer.

> **Important**
>
> Untuk menjalankan komponen ini dalam sebuah kontainer, Anda harus menentukan perangkat sistem yang Anda konfigurasi di variabel lingkungan `ModbusLocalPort`.

Objek ini berisi informasi berikut:

`0` - Ini adalah indeks himpunan sebagai string.

Objek yang berisi informasi berikut:

`path`

Jalur ke perangkat sistem pada perangkat inti. Ini harus memiliki nilai yang sama dengan nilai yang Anda konfigurasi untuk `ModbusLocalPort`.

`permission`

(Opsional) Izin untuk mengakses perangkat sistem dari kontainer. Nilai ini harus `rw`, yang menentukan bahwa komponen memiliki akses baca/tulis ke perangkat sistem.

Default: `rw`

`addGroupOwner`

(Opsional) Apakah akan menambahkan grup sistem atau tidak yang akan menjalankan komponen sebagai pemilik perangkat sistem.

Default: `true`

`pubsubTopics`

(Opsional) Sebuah objek yang berisi topik di mana komponen berlangganan untuk menerima pesan. Anda dapat menentukan setiap topik dan apakah komponen berlangganan topik MQTT dari AWS IoT Core atau topik penerbitan/langganan lokal.

Objek ini berisi informasi berikut:

`0` - Ini adalah indeks himpunan sebagai string.

Objek yang berisi informasi berikut:

`type`

(Opsional) Jenis olahpesan publikasikan/berlangganan yang digunakan oleh komponen ini untuk berlangganan pesan. Pilih dari salah satu pilihan berikut:

- `PUB_SUB` — Berlangganan pesan publish/subscribe lokal. Jika Anda memilih opsi ini, topik tidak dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Pesan lokal publikasi/berlangganan.
- `IOT_CORE`— Berlangganan pesan AWS IoT Core MQTT. Jika Anda memilih opsi ini, topik dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Terbitkan/berlangganan pesan MQTT AWS IoT Core.

Default: `PUB_SUB`

`topic`

(Opsional) Topik yang menjadi langganan komponen untuk menerima pesan. Jika Anda menentukan `IotCore` untuk `type`, Anda dapat menggunakan wildcard MQTT (+ dan #) dalam topik ini.

Example Contoh: Pembaruan gabungan konfigurasi (mode kontainer)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "GreengrassContainer",
```

```
  "containerParams": {
    "devices": {
      "0": {
        "path": "/dev/ttyS2",
        "permission": "rw",
        "addGroupOwner": true
      }
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi (tidak ada mode kontainer)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "NoContainer"
}
```

Data input

Komponen ini menerima parameter permintaan Modbus RTU pada topik berikut dan mengirimkan permintaan Modbus RTU ke perangkat. Secara default, komponen ini berlangganan topik publikasi/berlangganan lokal. Untuk informasi lebih lanjut tentang cara mempublikasikan pesan ke komponen ini dari komponen kustom Anda, lihat Pesan lokal publikasi/berlangganan.

Topik default (publish/subscribe lokal): `modbus/adapter/request`

Pesan menerima properti berikut. Pesan input harus dalam format JSON.

`request`

Parameter untuk permintaan Modbus RTU yang akan dikirim.

Bentuk pesan permintaan tergantung pada jenis Modbus RTU permintaan yang diwakilinya. Properti berikut diperlukan untuk semua permintaan.

Jenis: `object` yang berisi informasi berikut:

`operation`

Nama operasi yang akan dijalankan. Sebagai contoh, tentukan `ReadCoilsRequest` untuk membaca kumparan pada perangkat Modbus RTU. Untuk informasi lebih lanjut tentang operasi tulis, lihat Permintaan dan tanggapan Modbus RTU.

Tipe: `string`

`device`

Perangkat target permintaan.

Nilai ini harus berupa bilangan bulat antara `0` dan `247`.

Tipe: `integer`

Parameter lain yang akan tercakup dalam permintaan tergantung pada operasi. Komponen ini menangani pemeriksaan redundansi siklik (CRC) untuk memverifikasi permintaan data untuk Anda.

> **Note**
>
> Jika permintaan Anda mencakup properti `address`, Anda harus menentukan nilainya sebagai bilangan bulat. Misalnya, `"address": 1`.

`id`

ID acak untuk permintaan. Gunakan properti ini untuk memetakan permintaan inpun untuk respons output. Ketika Anda menentukan properti ini, komponen menetapkan properti `id` di objek respons untuk nilai ini.

Tipe: `string`

Example Contoh input: Baca permintaan kumparan

```
{
  "request": {
    "operation": "ReadCoilsRequest",
    "device": 1,
```

```
    "address": 1,
    "count": 1
  },
  "id": "MyRequest"
}
```

Data output

Komponen ini menerbitkan tanggapan sebagai data output pada topik MQTT berikut secara default. Anda harus menentukan topik ini sebagai `subject` dalam konfigurasi untuk komponen router langganan warisan. Untuk informasi lebih lanjut tentang cara mempublikasikan pesan ke komponen ini dari komponen kustom Anda, lihat Terbitkan/berlangganan pesan MQTT AWS IoT Core.

Topik default (AWS IoT Core MQTT): `modbus/adapter/response`

Bentuk pesan respon tergantung pada operasi permintaan dan status respon. Sebagai contoh, lihat Contoh permintaan dan respons.

Setiap respons mencakup properti berikut:

`response`

Respons dari perangkat Modbus RTU.

Jenis: `object` yang berisi informasi berikut:

`status`

Status HTTP dari permintaan. Nilai bisa jadi salah satu dari yang berikut:

- `Success` — Permintaannya valid, komponen mengirim permintaan ke jaringan Modbus RTU, dan jaringan RTU Modbus mengembalikan respons.
- `Exception` — Permintaannya valid, komponen mengirim permintaan ke jaringan Modbus RTU, dan jaringan RTU Modbus mengembalikan pengecualian. Untuk informasi selengkapnya, lihat Status respons: Pengecualian.
- `No Response` — Permintaan tidak valid, dan komponen menangkap kesalahan sebelum mengirim permintaan ke jaringan Modbus RTU. Untuk informasi selengkapnya, lihat Status respon: Tidak ada respon.

`operation`

Operasi yang diminta komponen.

`device`

Perangkat tempat komponen mengirim permintaan.

`payload`

Respons dari perangkat Modbus RTU. Jika `status` adalah `No Response`, objek ini hanya akan berisi properti `error` dengan deskripsi kesalahan (misalnya, `[Input/Output] No Response received from the remote unit`).

`id`

ID dari permintaan, yang dapat Anda gunakan untuk mengidentifikasi respons yang sesuai dengan yang diminta.

> **Note**
>
> Sebuah respons untuk operasi write hanyalah pengulangan dari permintaan. Meskipun tanggapan write tidak mencakup informasi yang berarti, ia merupakan praktik yang baik untuk memeriksa status respons untuk melihat apakah permintaan berhasil atau gagal.

Example Contoh output: Berhasil

```
{
  "response" : {
    "status" : "success",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "function_code": 1,
      "bits": [1]
    }
  },
  "id" : "MyRequest"
}
```

Example Contoh output: Gagal

```
{
  "response" : {
    "status" : "fail",
```

```
    "error_message": "Internal Error",
    "error": "Exception",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "function_code": 129,
      "exception_code": 2
    }
  },
  "id" : "MyRequest"
}
```

Untuk contoh lainnya, lihat 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 kumparan

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 yang memiliki

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,
    "count": 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"
}
```

Tulis register tunggal

Contoh permintaan:

```
{
  "request": {
    "operation": "WriteSingleRegisterRequest",
    "device": 1,
    "address": 1,
    "value": 1
  },
  "id": "TestRequest"
}
```

Tulis beberapa kumparan

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 beberapa 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"
}
```

Register mask write

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 beberapa 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**
>
> Tanggapan meliputi register yang dibaca komponen.

Status respons: 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:

- `status` ditetapkan ke `No Response`.
- `error` berisi alasan kesalahan.
- `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 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"
}
```

File log lokal

Komponen ini menggunakan file log berikut.

```
/greengrass/v2/logs/aws.greengrass.Modbus.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* dengan jalur ke folder AWS IoT Greengrass root.

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

Lisensi

Komponen ini mencakup perangkat lunak/lisensi pihak ketiga berikut:

- pymodbusLisensi BSD
- pyserialLisensi BSD

Komponen ini dirilis menurut Perjanjian Lisensi Perangkat Lunak Greengrass Core.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.8	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.

Versi	Perubahan
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.5	Perbaikan bug dan peningkatan • Memperbaiki masalah dengan `ReadDiscreteInput` operasi.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.2	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.1	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.0	Fitur baru • Menambahkan`ModbusBaudRate` ,, `ModbusByteSize` `ModbusParity` , dan `ModbusStopBits` opsi yang dapat Anda tentukan untuk mengonfigurasi komunikasi serial dengan perangkat Modbus RTU.
2.0.8	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.7	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.6	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.5	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.0.4	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.
2.0.3	Versi awal.

Jembatan MQTT

Komponen jembatan MQTT (`aws.greengrass.clientdevices.mqtt.Bridge`) menyampaikan pesan MQTT antara perangkat klien, penerbitan/berlangganan Greengrass lokal, dan. AWS IoT Core

Anda dapat menggunakan komponen ini untuk bertindak atas pesan MQTT dari perangkat klien komponen kustom dan mensinkronisasi perangkat klien dengan AWS Cloud.

> **Note**
>
> Perangkat klien adalah perangkat IoT lokal yang terhubung ke perangkat inti Greengrass untuk mengirim pesan MQTT dan data yang akan diproses. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Anda dapat menggunakan komponen ini untuk merelai pesan antara broker-broker pesan berikut:

- MQTT lokal - Broker MQTT lokal menangani pesan antara perangkat klien dan perangkat inti.
- Publikasi/berlangganan lokal - Broker pesan Greengrass lokal menangani pesan antar komponen pada perangkat inti. Untuk informasi lebih lanjut tentang cara berinteraksi dengan pesan-pesan ini dalam komponen Greengrass, lihat.Pesan lokal publikasi/berlangganan.
- AWS IoT Core — Pialang AWS IoT Core MQTT menangani pesan antara perangkat dan tujuan IoT. AWS Cloud Untuk informasi lebih lanjut tentang cara berinteraksi dengan pesan-pesan ini dalam komponen Greengrass, lihat.Terbitkan/berlangganan pesan MQTT AWS IoT Core.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

Topik

- Versi
- Jenis
- Sistem operasi
- Persyaratan
- Dependensi

- Konfigurasi
- Berkas log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Jenis

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Jika Anda mengonfigurasi komponen broker MQTT perangkat inti untuk menggunakan port selain port default 8883, Anda harus menggunakan jembatan MQTT v2.1.0 atau yang lebih baru. Konfigurasikan untuk terhubung di port tempat broker beroperasi.

- Komponen jembatan MQTT didukung untuk berjalan di VPC.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.3.2

Tabel berikut mencantumkan dependensi untuk versi 2.3.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.6.0	Keras

2.3.0 and 2.3.1

Tabel berikut mencantumkan dependensi untuk versi 2.3.0 dan 2.3.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.5.0	Keras

2.2.5 and 2.2.6

Tabel berikut mencantumkan dependensi untuk versi 2.2.5 dan 2.2.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.5.0	Keras

2.2.3 and 2.2.4

Tabel berikut mencantumkan dependensi untuk versi 2.2.3 dan 2.2.4 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.4.0	Keras

2.2.0 – 2.2.2

Tabel berikut mencantumkan dependensi untuk versi 2.2.0 hingga 2.2.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.3.0	Keras

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.0.0 <2.2.0	Keras

2.0.0 to 2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.0 hingga 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.0.0 <2.1.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

2.3.0 – 2.3.2

`mqttTopicMapping`

Pemetaan topik yang ingin Anda jembatani. Komponen ini berlangganan pesan pada topik sumber dan menerbitkan pesan yang diterimanya ke topik tujuan. Setiap pemetaan topik menentukan topik, jenis sumber, dan jenis tujuan.

Objek ini berisi informasi berikut:

`topicMappingNameKey`

Nama pemetaan topik ini. Ganti *`topicMappingNameKunci`* dengan nama yang membantu Anda mengidentifikasi pemetaan topik ini.

Objek ini berisi informasi berikut:

`topic`

Filter topik atau topik untuk menjembatani antara broker sumber dan target.

Anda dapat menggunakan wildcard topik + dan # MQTT untuk menyampaikan pesan pada semua topik yang cocok dengan filter topik. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

> **Note**
>
> Untuk menggunakan wildcard topik MQTT dengan broker `Pubsub` sumber, Anda harus menggunakan v2.6.0 atau yang lebih baru dari komponen inti Greengrass.

`targetTopicPrefix`

Awalan untuk ditambahkan ke topik target saat komponen ini menyampaikan pesan.

`source`

Broker pesan sumber. Pilih dari salah satu pilihan berikut:

- `LocalMqtt` – Broker MQTT lokal tempat perangkat klien berkomunikasi.
- `Pubsub` – Broker pesan publikasi/berlangganan Greengrass lokal.
- `IotCore`— Pialang AWS IoT Core pesan MQTT.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

`source` dan `target` harus berbeda.

`target`

Target broker pesan. Pilih dari salah satu pilihan berikut:

- `LocalMqtt` – Broker MQTT lokal tempat perangkat klien berkomunikasi.
- `Pubsub` – Broker pesan publikasi/berlangganan Greengrass lokal.
- `IotCore`— Pialang AWS IoT Core pesan MQTT.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

`source` dan `target` harus berbeda.

mqtt5 RouteOptions

(Opsional) Menyediakan opsi untuk mengonfigurasi pemetaan topik untuk menjembatani pesan dari topik sumber ke topik tujuan.

Objek ini berisi informasi berikut:

mqtt5 RouteOptionsNameKey

Nama opsi rute untuk pemetaan topik. Ganti *mqtt5 RouteOptionsNameKey* dengan *topicMappingNameKunci* yang cocok yang ditentukan di lapangan. `mqttTopicMapping`

Objek ini berisi informasi berikut:

NoLokal

(Opsional) Saat diaktifkan, bridge tidak meneruskan pesan pada topik yang dipublikasikan oleh bridge itu sendiri. Gunakan ini untuk mencegah loop, sebagai berikut:

```
{
    "mqtt5RouteOptions": {
        "toIoTCore": {
            "noLocal": true
        }
    },
    "mqttTopicMapping": {
        "toIoTCore": {
            "topic": "device",
            "source": "LocalMqtt",
            "target": "IotCore"
        },
        "toLocal": {
            "topic": "device",
            "source": "IotCore",
            "target": "LocalMqtt"
        }
    }
}
```

`noLocal`hanya didukung untuk rute di mana `source` berada`LocalMqtt`.

Bawaan: salah

retainAsPublished

(Opsional) Saat diaktifkan, pesan yang diteruskan oleh jembatan memiliki `retain` tanda yang sama dengan pesan yang dipublikasikan ke broker untuk rute tersebut.

`retainAsPublished`hanya didukung untuk rute di mana `source` berada`LocalMqtt`.

Bawaan: salah

mqtt

(Opsional) Pengaturan protokol MQTT untuk berkomunikasi dengan broker lokal.

versi

(Opsional) Versi protokol MQTT yang digunakan oleh jembatan untuk berkomunikasi dengan broker lokal. Harus sama dengan versi MQTT yang dipilih dalam konfigurasi nukleus.

Pilih dari yang berikut ini:

- `mqtt3`
- `mqtt5`

Anda harus menggunakan broker MQTT ketika `target` bidang `source` atau `mqttTopicMapping` objek diatur ke. `LocalMqtt` Jika Anda memilih `mqtt5` opsi, Anda harus menggunakanPialang MQTT 5 (EMQX).

Default: `mqtt3`

ackTimeoutSeconds

(Opsional) Interval waktu untuk menunggu paket PUBACK, SUBACK, atau UNSUBACK sebelum gagal operasi.

Default: 60

connAckTimeoutNona

(Opsional) Interval waktu untuk menunggu paket CONNACK sebelum mematikan koneksi.

Default: 20000 (20 detik)

pingTimeoutMs

(Opsional) Jumlah waktu dalam milidetik jembatan menunggu untuk menerima pesan PINGACK dari broker lokal. Jika menunggu melebihi batas waktu, jembatan

ditutup kemudian membuka kembali koneksi MQTT. Nilai ini harus kurang dari`keepAliveTimeoutSeconds`.

Default: 30000 (30 detik)

keepAliveTimeoutDetik

(Opsional) Jumlah waktu dalam detik antara setiap pesan PING yang dikirim jembatan untuk menjaga koneksi MQTT tetap hidup. Nilai ini harus lebih besar dari`pingTimeoutMs`.

Default: 60

maxReconnectDelayNona

(Opsional) Jumlah waktu maksimum dalam hitungan detik agar MQTT terhubung kembali.

Default: 30000 (30 detik)

minReconnectDelayNona

(Opsional) Jumlah waktu minimum dalam hitungan detik agar MQTT terhubung kembali.

ReceiveMaximum

(Opsional) Jumlah maksimum paket QoS1 yang tidak diakui yang dapat dikirim oleh jembatan.

Default: 100

maximumPacketSize

Jumlah maksimum byte yang akan diterima klien untuk paket MQTT.

Default: null (Tidak ada batas)

sessionExpiryInterval

(Opsional) Jumlah waktu dalam hitungan detik Anda dapat meminta sesi berlangsung antara jembatan dan broker lokal.

Default: 4294967295 (sesi tidak pernah kedaluwarsa)

`brokerUri`

(Opsional) URI dari broker MQTT lokal. Anda harus menentukan parameter ini jika Anda mengonfigurasi broker MQTT untuk menggunakan port yang berbeda dari port default 8883. Gunakan format berikut, dan ganti *`port`* dengan port tempat broker MQTT beroperasi:. `ssl://localhost:`*`port`*

Default: `ssl://localhost:8883`

startupTimeoutSeconds

(Opsional) Maksimum waktu dalam hitungan detik untuk memulai komponen. Status komponen berubah menjadi `BROKEN` jika melebihi batas waktu ini.

Default: `120`

Example Contoh: Pembaruan gabungan konfigurasi

Berikut contoh update konfigurasi menentukan berikut:

- Relay pesan dari perangkat klien ke AWS IoT Core topik yang cocok dengan filter `clients/+/hello/world` topik.
- Relay pesan dari perangkat klien ke publish/subscribe lokal pada topik yang cocok dengan filter `clients/+/detections` topik, dan tambahkan `events/input/` awalan ke topik target. Topik target yang dihasilkan cocok dengan filter `events/input/clients/+/detections` topik.
- Relay pesan dari perangkat klien ke AWS IoT Core topik yang cocok dengan filter `clients/+/status` topik, dan tambahkan `$aws/rules/StatusUpdateRule/` awalan ke topik target. Contoh ini menyampaikan pesan-pesan ini langsung ke AWS IoT aturan bernama `StatusUpdateRule` untuk mengurangi biaya menggunakan Basic Ingest.

```
{
  "mqttTopicMapping": {
    "ClientDeviceHelloWorld": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDeviceEvents": {
      "topic": "clients/+/detections",
      "targetTopicPrefix": "events/input/",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ClientDeviceCloudStatusUpdate": {
      "topic": "clients/+/status",
      "targetTopicPrefix": "$aws/rules/StatusUpdateRule/",
```

```
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

Example Contoh: Mengkonfigurasi MQTT 5

Contoh konfigurasi berikut memperbarui yang berikut ini:

- Memungkinkan jembatan untuk menggunakan protokol MQTT 5 dengan broker lokal.
- Mengkonfigurasi penyimpanan MQTT sebagai pengaturan yang dipublikasikan untuk pemetaan topik. `ClientDeviceHelloWorld`

```
{
  "mqttTopicMapping": {
    "ClientDeviceHelloWorld": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  },
  "mqtt5RouteOptions": {
    "ClientDeviceHelloWorld": {
      "retainAsPublished": true
    }
  },
  "mqtt": {
    "version": "mqtt5"
  }
}
```

2.2.6

`mqttTopicMapping`

Pemetaan topik yang ingin Anda jembatani. Komponen ini berlangganan pesan pada topik sumber dan menerbitkan pesan yang diterimanya ke topik tujuan. Setiap pemetaan topik menentukan topik, jenis sumber, dan jenis tujuan.

Objek ini berisi informasi berikut:

`topicMappingNameKey`

Nama pemetaan topik ini. Ganti *`topicMappingNameKunci`* dengan nama yang membantu Anda mengidentifikasi pemetaan topik ini.

Objek ini berisi informasi berikut:

`topic`

Filter topik atau topik untuk menjembatani antara broker sumber dan target.

Anda dapat menggunakan wildcard topik + dan # MQTT untuk menyampaikan pesan pada semua topik yang cocok dengan filter topik. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

> **Note**
>
> Untuk menggunakan wildcard topik MQTT dengan broker `Pubsub` sumber, Anda harus menggunakan v2.6.0 atau yang lebih baru dari komponen inti Greengrass.

`targetTopicPrefix`

Awalan untuk ditambahkan ke topik target saat komponen ini menyampaikan pesan.

`source`

Broker pesan sumber. Pilih dari salah satu pilihan berikut:

- `LocalMqtt` – Broker MQTT lokal tempat perangkat klien berkomunikasi.
- `Pubsub` – Broker pesan publikasi/berlangganan Greengrass lokal.
- `IotCore`— Pialang AWS IoT Core pesan MQTT.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core

Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

`source` dan `target` harus berbeda.

`target`

Target broker pesan. Pilih dari salah satu pilihan berikut:

- `LocalMqtt` – Broker MQTT lokal tempat perangkat klien berkomunikasi.
- `Pubsub` – Broker pesan publikasi/berlangganan Greengrass lokal.
- `IotCore`— Pialang AWS IoT Core pesan MQTT.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

`source` dan `target` harus berbeda.

`brokerUri`

(Opsional) URI dari broker MQTT lokal. Anda harus menentukan parameter ini jika Anda mengonfigurasi broker MQTT untuk menggunakan port yang berbeda dari port default 8883. Gunakan format berikut, dan ganti *`port`* dengan port tempat broker MQTT beroperasi:. `ssl://localhost:`*`port`*

Default: `ssl://localhost:8883`

startupTimeoutSeconds

(Opsional) Maksimum waktu dalam hitungan detik untuk memulai komponen. Status komponen berubah menjadi `BROKEN` jika melebihi batas waktu ini.

Default: `120`

Example Contoh: Pembaruan gabungan konfigurasi

Berikut contoh update konfigurasi menentukan berikut:

- Relay pesan dari perangkat klien ke AWS IoT Core topik yang cocok dengan filter `clients/+/hello/world` topik.
- Relay pesan dari perangkat klien ke publish/subscribe lokal pada topik yang cocok dengan filter `clients/+/detections` topik, dan tambahkan `events/input/` awalan ke topik target. Topik target yang dihasilkan cocok dengan filter `events/input/clients/+/detections` topik.
- Relay pesan dari perangkat klien ke AWS IoT Core topik yang cocok dengan filter `clients/+/status` topik, dan tambahkan `$aws/rules/StatusUpdateRule/` awalan ke topik target. Contoh ini menyampaikan pesan-pesan ini langsung ke AWS IoT aturan bernama `StatusUpdateRule` untuk mengurangi biaya menggunakan Basic Ingest.

```
{
  "mqttTopicMapping": {
    "ClientDeviceHelloWorld": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDeviceEvents": {
      "topic": "clients/+/detections",
      "targetTopicPrefix": "events/input/",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ClientDeviceCloudStatusUpdate": {
      "topic": "clients/+/status",
      "targetTopicPrefix": "$aws/rules/StatusUpdateRule/",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

2.2.0 - 2.2.5

`mqttTopicMapping`

Pemetaan topik yang ingin Anda jembatani. Komponen ini berlangganan pesan pada topik sumber dan menerbitkan pesan yang diterimanya ke topik tujuan. Setiap pemetaan topik menentukan topik, jenis sumber, dan jenis tujuan.

Objek ini berisi informasi berikut:

`topicMappingNameKey`

Nama pemetaan topik ini. Ganti *`topicMappingNameKunci`* dengan nama yang membantu Anda mengidentifikasi pemetaan topik ini.

Objek ini berisi informasi berikut:

`topic`

Filter topik atau topik untuk menjembatani antara broker sumber dan target.

Anda dapat menggunakan wildcard topik + dan # MQTT untuk menyampaikan pesan pada semua topik yang cocok dengan filter topik. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

> **Note**
>
> Untuk menggunakan wildcard topik MQTT dengan broker `Pubsub` sumber, Anda harus menggunakan v2.6.0 atau yang lebih baru dari komponen inti Greengrass.

`targetTopicPrefix`

Awalan untuk ditambahkan ke topik target saat komponen ini menyampaikan pesan.

`source`

Broker pesan sumber. Pilih dari salah satu pilihan berikut:

- `LocalMqtt` – Broker MQTT lokal tempat perangkat klien berkomunikasi.
- `Pubsub` – Broker pesan publikasi/berlangganan Greengrass lokal.
- `IotCore`— Pialang AWS IoT Core pesan MQTT.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

`source` dan `target` harus berbeda.

`target`

Target broker pesan. Pilih dari salah satu pilihan berikut:

- `LocalMqtt` – Broker MQTT lokal tempat perangkat klien berkomunikasi.
- `Pubsub` – Broker pesan publikasi/berlangganan Greengrass lokal.
- `IotCore`— Pialang AWS IoT Core pesan MQTT.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

`source` dan `target` harus berbeda.

`brokerUri`

(Opsional) URI dari broker MQTT lokal. Anda harus menentukan parameter ini jika Anda mengonfigurasi broker MQTT untuk menggunakan port yang berbeda dari port default 8883. Gunakan format berikut, dan ganti *`port`* dengan port tempat broker MQTT beroperasi:. `ssl://localhost:`*`port`*

Default: `ssl://localhost:8883`

Example Contoh: Pembaruan gabungan konfigurasi

Berikut contoh update konfigurasi menentukan berikut:

- Relay pesan dari perangkat klien ke AWS IoT Core topik yang cocok dengan filter `clients/+/hello/world` topik.
- Relay pesan dari perangkat klien ke publish/subscribe lokal pada topik yang cocok dengan filter `clients/+/detections` topik, dan tambahkan `events/input/` awalan ke topik target. Topik target yang dihasilkan cocok dengan filter `events/input/clients/+/detections` topik.
- Relay pesan dari perangkat klien ke AWS IoT Core topik yang cocok dengan filter `clients/+/status` topik, dan tambahkan `$aws/rules/StatusUpdateRule/` awalan ke topik target. Contoh ini menyampaikan pesan-pesan ini langsung ke AWS IoT aturan bernama `StatusUpdateRule` untuk mengurangi biaya menggunakan Basic Ingest.

```
{
  "mqttTopicMapping": {
    "ClientDeviceHelloWorld": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDeviceEvents": {
      "topic": "clients/+/detections",
      "targetTopicPrefix": "events/input/",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ClientDeviceCloudStatusUpdate": {
      "topic": "clients/+/status",
      "targetTopicPrefix": "$aws/rules/StatusUpdateRule/",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

2.1.x

`mqttTopicMapping`

Pemetaan topik yang ingin Anda jembatani. Komponen ini berlangganan pesan pada topik sumber dan menerbitkan pesan yang diterimanya ke topik tujuan. Setiap pemetaan topik menentukan topik, jenis sumber, dan jenis tujuan.

Objek ini berisi informasi berikut:

`topicMappingNameKey`

Nama pemetaan topik ini. Ganti *`topicMappingNameKunci`* dengan nama yang membantu Anda mengidentifikasi pemetaan topik ini.

Objek ini berisi informasi berikut:

`topic`

Filter topik atau topik untuk menjembatani antara broker sumber dan target.

Jika Anda menentukan broker sumber `LocalMqtt` atau `IotCore`, Anda dapat menggunakan wildcard topik MQTT + dan # untuk merelai pesan pada semua topik yang cocok dengan filter topik. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

`source`

Broker pesan sumber. Pilih dari salah satu pilihan berikut:

- `LocalMqtt` – Broker MQTT lokal tempat perangkat klien berkomunikasi.
- `Pubsub` – Broker pesan publikasi/berlangganan Greengrass lokal.
- `IotCore`— Pialang AWS IoT Core pesan MQTT.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core

Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

`source` dan `target` harus berbeda.

`target`

Target broker pesan. Pilih dari salah satu pilihan berikut:

- `LocalMqtt` – Broker MQTT lokal tempat perangkat klien berkomunikasi.
- `Pubsub` – Broker pesan publikasi/berlangganan Greengrass lokal.
- `IotCore`— Pialang AWS IoT Core pesan MQTT.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

`source` dan `target` harus berbeda.

`brokerUri`

(Opsional) URI dari broker MQTT lokal. Anda harus menentukan parameter ini jika Anda mengonfigurasi broker MQTT untuk menggunakan port yang berbeda dari port default 8883. Gunakan format berikut, dan ganti *`port`* dengan port tempat broker MQTT beroperasi:. `ssl://localhost:`*`port`*

Default: `ssl://localhost:8883`

Example Contoh: Pembaruan gabungan konfigurasi

Contoh pembaruan konfigurasi berikut menentukan untuk menyampaikan pesan dari perangkat klien ke AWS IoT Core topik `clients/MyClientDevice1/hello/world` dan `clients/MyClientDevice2/hello/world` topik.

```
{
  "mqttTopicMapping": {
    "ClientDevice1HelloWorld": {
      "topic": "clients/MyClientDevice1/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDevice2HelloWorld": {
      "topic": "clients/MyClientDevice2/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

2.0.x

`mqttTopicMapping`

Pemetaan topik yang ingin Anda jembatani. Komponen ini berlangganan pesan pada topik sumber dan menerbitkan pesan yang diterimanya ke topik tujuan. Setiap pemetaan topik menentukan topik, jenis sumber, dan jenis tujuan.

Objek ini berisi informasi berikut:

`topicMappingNameKey`

Nama pemetaan topik ini. Ganti *`topicMappingNameKunci`* dengan nama yang membantu Anda mengidentifikasi pemetaan topik ini.

Objek ini berisi informasi berikut:

`topic`

Filter topik atau topik untuk menjembatani antara broker sumber dan target.

Jika Anda menentukan broker sumber `LocalMqtt` atau `IotCore`, Anda dapat menggunakan wildcard topik MQTT + dan # untuk merelai pesan pada semua topik yang cocok dengan filter topik. Untuk informasi selengkapnya, lihat topik MQTT di Panduan Developer AWS IoT Core .

`source`

Broker pesan sumber. Pilih dari salah satu pilihan berikut:

- `LocalMqtt` – Broker MQTT lokal tempat perangkat klien berkomunikasi.
- `Pubsub` – Broker pesan publikasi/berlangganan Greengrass lokal.
- `IotCore`— Pialang AWS IoT Core pesan MQTT.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

`source` dan `target` harus berbeda.

`target`

Target broker pesan. Pilih dari salah satu pilihan berikut:

- `LocalMqtt` – Broker MQTT lokal tempat perangkat klien berkomunikasi.
- `Pubsub` – Broker pesan publikasi/berlangganan Greengrass lokal.
- `IotCore`— Pialang AWS IoT Core pesan MQTT.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Akibatnya, Anda mungkin mengamati latensi tambahan saat menyampaikan pesan MQTT dari perangkat klien di broker MQTT lokal ke. AWS IoT Core Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

`source` dan `target` harus berbeda.

Example Contoh: Pembaruan gabungan konfigurasi

Contoh pembaruan konfigurasi berikut menentukan untuk menyampaikan pesan dari perangkat klien ke AWS IoT Core topik `clients/MyClientDevice1/hello/world` dan `clients/MyClientDevice2/hello/world` topik.

```
{
  "mqttTopicMapping": {
    "ClientDevice1HelloWorld": {
      "topic": "clients/MyClientDevice1/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDevice2HelloWorld": {
      "topic": "clients/MyClientDevice2/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

Berkas log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

Linux

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

Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.3.2	Versi diperbarui untuk perangkat klien autentikasi versi 2.5.0 rilis.
2.3.1	Perbaikan bug dan peningkatan Memperbaiki masalah saat klien MQTT lokal masuk ke loop pemutusan.
2.3.0	Fitur baru Menambahkan dukungan MQTT5 untuk menjembatani antara AWS IoT Core dan sumber MQTT lokal.
2.2.6	Fitur baru Menambahkan opsi `startupTimeoutSeconds` konfigurasi baru.
2.2.5	Versi diperbarui untuk perangkat klien autentikasi versi 2.4.0 rilis.
2.2.4	Versi diperbarui untuk Greengrass perangkat klien auth versi 2.3.0 rilis.
2.2.3	Versi ini berisi perbaikan bug dan perbaikan.
2.2.2	Perbaikan bug dan peningkatan • Penyesuaian logging.

Versi	Perubahan
2.2.1	Perbaikan bug dan peningkatan Memperbaiki masalah yang dapat mengakibatkan jembatan MQTT gagal berlangganan topik MQTT.
2.2.0	Fitur baru • Menambahkan dukungan untuk wildcard topik MQTT (#dan+) saat Anda menentukan publish/subscribe lokal sebagai broker pesan sumber. Fitur ini membutuhkan v2.6.0 atau yang lebih baru dari komponen inti Greengrass. • Menambahkan `targetTopicPrefix` opsi, yang dapat Anda tentukan untuk mengonfigurasi jembatan MQTT untuk menambahkan awalan ke topik target saat menyampaikan pesan.
2.1.1	Perbaikan bug dan peningkatan • Memperbaiki masalah dengan cara komponen ini menangani pembaruan pengaturan ulang konfigurasi. • Mengurangi frekuensi pemutusan klien MQTT saat sertifikat diputar.
2.1.0	Fitur baru • Menambahkan `brokerUri` parameter, yang memungkinkan Anda untuk menggunakan port broker MQTT non-default.
2.0.1	Versi ini mencakup perbaikan bug dan peningkatan.
2.0.0	Versi awal.

MQTT 3.1.1 broker (Moquette)

Komponen broker Moquette MQTT (`aws.greengrass.clientdevices.mqtt.Moquette`) menangani pesan MQTT antara perangkat klien dan perangkat inti Greengrass. Komponen ini menyediakan versi modifikasi broker Moquette MQTT. Sebarkan broker MQTT ini untuk menjalankan broker MQTT ringan. Untuk informasi lebih lanjut tentang cara memilih broker MQTT, lihat. Pilih broker MQTT

Broker ini mengimplementasikan protokol MQTT 3.1.1. Ini termasuk dukungan untuk QoS 0, QoS 1, pesan yang dipertahankan QoS 2, pesan kehendak terakhir, dan sesi persisten.

> **Note**
>
> Perangkat klien adalah perangkat IoT lokal yang terhubung ke perangkat inti Greengrass untuk mengirim pesan MQTT dan data yang akan diproses. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Jenis

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti harus dapat menerima koneksi pada port tempat broker MQTT beroperasi. Komponen ini menjalankan broker MQTT pada port 8883 secara default. Anda dapat menentukan port yang berbeda saat Anda mengonfigurasi komponen ini.

  Jika Anda menentukan port yang berbeda, dan Anda menggunakan komponen jembatan MQTT untuk menyampaikan pesan MQTT ke broker lain, Anda harus menggunakan MQTT bridge v2.1.0 atau yang lebih baru. Konfigurasikan untuk menggunakan port tempat broker MQTT beroperasi.

  Jika Anda menentukan port yang berbeda, dan Anda menggunakan komponen detektor IP untuk mengelola titik akhir broker MQTT, Anda harus menggunakan detektor IP v2.1.0 atau yang lebih baru. Konfigurasikan untuk melaporkan port tempat broker MQTT beroperasi.
- Komponen broker Moquette MQTT didukung untuk berjalan di VPC.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.3.7

Tabel berikut mencantumkan dependensi untuk versi 2.3.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.6.0	Keras

2.3.2 – 2.3.6

Tabel berikut mencantumkan dependensi untuk versi 2.3.2 hingga 2.3.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.5.0	Keras

2.3.0 and 2.3.1

Tabel berikut mencantumkan dependensi untuk versi 2.3.0 dan 2.3.1 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.4.0	Keras

2.2.0

Tabel berikut mencantumkan dependensi untuk versi 2.2.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.3.0	Keras

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.0.0 <2.2.0	Keras

2.0.0 - 2.0.2

Tabel berikut mencantumkan dependensi untuk versi 2.0.0 hingga 2.0.2 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.0.0 <2.1.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`moquette`

(Opsional) KOnfigurasi broker Moquette MQTT yang akan digunakan. Anda dapat mengonfigurasi subset dari opsi konfigurasi Moqeutte dalam komponen ini. Untuk informasi selengkapnya, lihat komentar sebaris pada file konfigurasi Moquette.

Objek ini berisi informasi berikut:

`ssl_port`

(Opsional) Port tempat broker MQTT beroperasi.

> **Note**
>
> Jika Anda menentukan port yang berbeda, dan Anda menggunakan komponen jembatan MQTT untuk menyampaikan pesan MQTT ke broker lain, Anda harus menggunakan MQTT bridge v2.1.0 atau yang lebih baru. Konfigurasikan untuk menggunakan port tempat broker MQTT beroperasi.
> Jika Anda menentukan port yang berbeda, dan Anda menggunakan komponen detektor IP untuk mengelola titik akhir broker MQTT, Anda harus menggunakan

detektor IP v2.1.0 atau yang lebih baru. Konfigurasikan untuk melaporkan port tempat broker MQTT beroperasi.

Default: `8883`

`host`

(Opsional) Antarmuka tempat broker MQTT terikat. Misalnya, Anda dapat mengubah parameter ini sehingga broker MQTT terikat hanya pada jaringan lokal tertentu.

Default: `0.0.0.0` (terikat pada semua antarmuka jaringan)

startupTimeoutSeconds

(Opsional) Maksimum waktu dalam hitungan detik untuk memulai komponen. Status komponen berubah menjadi `BROKEN` jika melebihi batas waktu ini.

Default: `120`

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk mengoperasikan broker MQTT pada port 443.

```
{
  "moquette": {
    "ssl_port": "443"
  }
}
```

Berkas log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.3.7	Versi diperbarui untuk perangkat klien autentikasi versi 2.5.0 rilis.
2.3.6	Perbaikan bug dan peningkatan • Perbaikan bug umum dan perbaikan.
2.3.5	Perbaikan bug dan peningkatan • Diperbarui Moquette ke versi 0.17.
2.3.4	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana klien mungkin mengalami kesalahan sesi yang tidak valid saat mengirim atau menerima pesan, karena ID klien duplikat. Masalah ini menyebabkan sesi klien ditutup.
2.3.3	Fitur baru Menambahkan opsi `startupTimeoutSeconds` konfigurasi baru.
2.3.2	Versi diperbarui untuk perangkat klien autentikasi versi 2.4.0 rilis.

Versi	Perubahan
2.3.1	Perbaikan bug dan peningkatan • Memperbaiki kondisi balapan di mana klien mungkin terputus setelah mencoba menyambung kembali, karena sesi tidak valid.
2.3.0	Menambahkan dukungan untuk rantai sertifikat.
2.2.0	Versi diperbarui untuk perangkat klien autentikasi versi 2.2.0 rilis.
2.1.0	Perbaikan bug dan peningkatan • Memperbarui komponen ini untuk menggunakan Moquette versi 0.16, yang meningkatkan kinerja dan mencakup beberapa peningkatan lainnya. • Memperbaiki masalah di mana sertifikat server MQTT lokal berputar lebih sering daripada yang dimaksudkan dalam skenario tertentu. Untuk menerapkan perbaikan ini, Anda juga harus menggunakan v2.1.0 atau yang lebih baru dari komponen autentikasi perangkat klien.
2.0.2	Perbaikan bug dan peningkatan • Meningkatkan ukuran pesan MQTT maksimum dari 8.092 byte menjadi 128 kilobyte. Batas payload pesan MQTT efektif sedikit kurang, karena batas ukuran pesan termasuk header pesan. • Menambahkan dukungan untuk nilai integer dalam `ssl_port` parameter.
2.0.1	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.0	Versi awal.

Pialang MQTT 5 (EMQX)

Komponen broker EMQX MQTT (`aws.greengrass.clientdevices.mqtt.EMQX`) menangani pesan MQTT antara perangkat klien dan perangkat inti Greengrass. Komponen ini menyediakan versi modifikasi dari broker EMQX MQTT 5.0. Menyebarkan broker MQTT ini untuk menggunakan fitur

MQTT 5 dalam komunikasi antara perangkat klien dan perangkat inti. Untuk informasi lebih lanjut tentang cara memilih broker MQTT, lihat. Pilih broker MQTT

Broker ini mengimplementasikan protokol MQTT 5.0. Ini mencakup dukungan untuk interval kedaluwarsa sesi dan pesan, properti pengguna, langganan bersama, alias topik, dan banyak lagi. MQTT 5 kompatibel dengan MQTT 3.1.1, jadi jika Anda menjalankan broker Moquette MQTT 3.1.1, Anda dapat menggantinya dengan broker EMQX MQTT 5, dan perangkat klien dapat terus terhubung dan beroperasi seperti biasa.

> **Note**
>
> Perangkat klien adalah perangkat IoT lokal yang terhubung ke perangkat inti Greengrass untuk mengirim pesan MQTT dan data yang akan diproses. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.0.x
- 1.2.x
- 1.1.x
- 1.0.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti harus dapat menerima koneksi pada port tempat broker MQTT beroperasi. Komponen ini menjalankan broker MQTT pada port 8883 secara default. Anda dapat menentukan port yang berbeda saat Anda mengonfigurasi komponen ini.

  Jika Anda menentukan port yang berbeda, dan Anda menggunakan komponen jembatan MQTT untuk menyampaikan pesan MQTT ke broker lain, Anda harus menggunakan MQTT bridge v2.1.0 atau yang lebih baru. Konfigurasikan untuk menggunakan port tempat broker MQTT beroperasi.

  Jika Anda menentukan port yang berbeda, dan Anda menggunakan komponen detektor IP untuk mengelola titik akhir broker MQTT, Anda harus menggunakan detektor IP v2.1.0 atau yang lebih baru. Konfigurasikan untuk melaporkan port tempat broker MQTT beroperasi.
- Pada perangkat inti Linux, Docker diinstal dan dikonfigurasi pada perangkat inti:
  - Docker Engine 1.9.1 atau yang lebih baru diinstal pada perangkat inti Greengrass. Versi 20.10 adalah versi terbaru yang diverifikasi untuk bekerja dengan perangkat lunak AWS IoT Greengrass Core. Anda harus menginstal Docker langsung pada perangkat inti sebelum Anda menyebarkan komponen yang menjalankan kontainer Docker.
  - Daemon Docker dimulai dan berjalan pada perangkat inti sebelum Anda men-deploy komponen ini.
  - Pengguna sistem yang menjalankan komponen ini harus memiliki izin root atau administrator. Atau, Anda dapat menjalankan komponen ini sebagai pengguna sistem dalam `docker` grup dan

mengonfigurasi `requiresPrivileges` opsi komponen ini `false` untuk menjalankan broker EQMX MQTT tanpa hak istimewa.

- Komponen broker EMQX MQTT didukung untuk berjalan di VPC.
- Komponen broker EMQX MQTT tidak didukung di platform. `armv7`

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.0.1

Tabel berikut mencantumkan dependensi untuk versi 2.0.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.6.0	Keras

2.0.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.5.0	Keras

1.2.2 – 1.2.3

Tabel berikut mencantumkan dependensi untuk versi 1.2.2 hingga 1.2.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.5.0	Keras

1.2.0 and 1.2.1

Tabel berikut mencantumkan dependensi untuk versi 1.2.0 dan 1.2.1 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.4.0	Keras

1.0.0 and 1.1.0

Tabel berikut mencantumkan dependensi untuk versi 1.0.0 dan 1.1.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Autentikasi perangkat klien	>=2.2.0 <2.3.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

2.0.0 - 2.0.1

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

> **Important**
>
> Jika Anda menggunakan versi 2 dari komponen broker MQTT 5 (EMQX), Anda harus memperbarui file konfigurasi Anda. File konfigurasi versi 1 tidak berfungsi dengan versi 2.

EMQXconfig

(Opsional) Konfigurasi broker EMQX MQTT untuk digunakan. Anda dapat mengatur opsi konfigurasi EMQX dalam komponen ini.

Saat Anda menggunakan broker EMQX, Greengrass menggunakan konfigurasi default. Konfigurasi ini digunakan kecuali Anda memodifikasinya menggunakan bidang ini.

Memodifikasi pengaturan konfigurasi berikut menyebabkan komponen broker EMQX dimulai ulang. Perubahan konfigurasi lainnya berlaku tanpa memulai ulang komponen.

- `emqxConfig/cluster`
- `emqxConfig/node`
- `emqxConfig/rpc`

> **Note**
>
> `aws.greengrass.clientdevices.mqtt.EMQX`memungkinkan Anda mengonfigurasi opsi yang sensitif terhadap keamanan. Ini termasuk pengaturan TLS, otentikasi, dan penyedia otorisasi. Kami merekomendasikan konfigurasi default yang menggunakan otentikasi TLS timbal balik dan penyedia autentikasi perangkat klien Greengrass.

Example Contoh: Konfigurasi default

Contoh berikut menunjukkan default yang ditetapkan untuk broker MQTT 5 (EMQX). Anda dapat mengganti pengaturan ini menggunakan pengaturan `emqxConfig` konfigurasi.

```
{
  "authorization": {
    "no_match": "deny",
    "sources": []
  },
  "node": {
    "cookie": "<placeholder>"
  },
  "listeners": {
     "ssl": {
       "default": {
         "ssl_options": {
           "keyfile": "{work:path}\\data\\key.pem",
           "certfile": "{work:path}\\data\\cert.pem",
           "cacertfile": null,
           "verify": "verify_peer",
           "versions": ["tlsv1.3", "tlsv1.2"],
           "fail_if_no_peer_cert": true
         }
       }
```

```
      },
      "tcp": {
        "default": {
          "enabled": false
        }
      },
      "ws": {
        "default": {
          "enabled": false
        }
      },
      "wss": {
        "default": {
          "enabled": false
        }
      }
    },
    "plugins": {
      "states": [{"name_vsn": "gg-1.0.0", "enable": true}],
      "install_dir": "plugins"
    }
}
```

AuthMode

(Opsional) Menetapkan penyedia otorisasi untuk broker. Dapat menjadi salah satu dari nilai berikut:

- `enabled`— (Default) Gunakan penyedia otentikasi dan otorisasi Greengrass.
- `bypass_on_failure`— Gunakan penyedia otentikasi Greengrass, lalu gunakan penyedia otentikasi yang tersisa di rantai penyedia EMQX jika Greengrass menolak otentikasi atau otorisasi.
- `bypass`— Penyedia Greengrass dinonaktifkan. Otentikasi dan otorisasi ditangani oleh rantai penyedia EMQX.

`requiresPrivilege`

(Opsional) Pada perangkat inti Linux, Anda dapat menentukan untuk menjalankan broker EMQX MQTT tanpa hak root atau administrator. Jika Anda menyetel opsi ini`false`, pengguna sistem yang menjalankan komponen ini harus menjadi anggota `docker` grup.

Default: `true`

`startupTimeoutSeconds`

(Opsional) Maksimum waktu dalam hitungan detik untuk memulai broker EMQX MQTT. Status komponen berubah menjadi `BROKEN` jika melebihi batas waktu ini.

Default: `90`

`ipcTimeoutSeconds`

(Opsional) Maksimum waktu dalam hitungan detik bagi komponen untuk menunggu inti Greengrass merespons permintaan komunikasi antarproses (IPC). Tingkatkan angka ini jika komponen ini melaporkan kesalahan batas waktu saat memeriksa apakah perangkat klien diotorisasi.

Default: `5`

`crtLogLevel`

(Opsional) Tingkat log untuk pustaka AWS Common Runtime (CRT).

Default ke level log broker EMQX MQTT (in). `log.level` `emqx`

`restartIdentifier`

(Opsional) Konfigurasikan opsi ini untuk memulai ulang broker EMQX MQTT. Ketika nilai konfigurasi ini berubah, komponen ini memulai ulang broker MQTT. Anda dapat menggunakan opsi ini untuk memaksa perangkat klien memutuskan sambungan.

`dockerOptions`

(Opsional) Konfigurasikan opsi ini hanya pada sistem operasi Linux untuk menambahkan parameter ke baris perintah Docker. Misalnya, untuk memetakan port tambahan, gunakan parameter `-p` Docker:

```
"-p 1883:1883"
```

Example Contoh: Memperbarui file konfigurasi v1.x ke v2.x

Contoh berikut menunjukkan perubahan yang diperlukan untuk memperbarui file konfigurasi v1.x ke versi 2.x.

File konfigurasi versi 1.x:

```
{
    "emqx": {
        "listener.ssl.external": "443",
        "listener.ssl.external.max_connections": "1024000",
        "listener.ssl.external.max_conn_rate": "500",
        "listener.ssl.external.rate_limit": "50KB,5s",
        "listener.ssl.external.handshake_timeout": "15s",
        "log.level": "warning"
    },
    "mergeConfigurationFiles": {
        "etc/plugins/aws_greengrass_emqx_auth.conf": "auth_mode=enabled\n
 use_greengrass_managed_certificates=true\n"
    }
}
```

File konfigurasi yang setara untuk v2:

```
{
    "emqxConfig": {
        "listeners": {
            "ssl": {
                "default": {
                   "bind": "8883",
                   "max_connections": "1024000",
                   "max_conn_rate": "500",
                   "handshake_timeout": "15s"
                }
            }
        },
        "log": {
            "console": {
              "enable": true,
              "level": "warning"
            }
        }
    },
    "authMode": "enabled"
}
```

Tidak ada yang setara dengan entri `listener.ssl.external.rate_limit` konfigurasi. Opsi `use_greengrass_managed_certificates` konfigurasi telah dihapus.

Example Contoh: Tetapkan port baru untuk broker

Contoh berikut mengubah port tempat broker MQTT beroperasi dari default 8883 ke port 1234. Jika Anda menggunakan Linux, sertakan `dockerOptions` bidangnya.

```
{
  "emqxConfig": {
    "listeners": {
      "ssl": {
        "default": {
          "bind": 1234
        }
      }
    }
  },
  "dockerOptions": "-p 1234:1234"
}
```

Example Contoh: Sesuaikan level log broker MQTT

Contoh berikut mengubah level log broker MQTT menjadi. `debug` Anda dapat memilih dari level log berikut:

- `debug`
- `info`
- `notice`
- `warning`
- `error`
- `critical`
- `alert`
- `emergency`

Level log default adalah`warning`.

```
{
  "emqxConfig": {
    "log": {
      "console": {
         "level": "debug"
```

```
      }
    }
  }
}
```

Example Contoh: Aktifkan dasbor EMQX

Contoh berikut memungkinkan dasbor EMQX sehingga Anda dapat memantau dan mengelola broker Anda. Jika Anda menggunakan Linux, sertakan `dockerOptions` bidangnya.

```
{
  "emqxConfig": {
    "dashboard": {
      "listeners": {
        "http": {
          "bind": 18083
        }
      }
    }
  },
  "dockerOptions": "-p 18083:18083"
}
```

1.0.0 - 1.2.2

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`emqx`

(Opsional) Konfigurasi broker EMQX MQTT untuk digunakan. Anda dapat mengonfigurasi subset opsi konfigurasi EMQX dalam komponen ini.

Objek ini berisi informasi berikut:

`listener.ssl.external`

(Opsional) Port tempat broker MQTT beroperasi.

> **Note**
>
> Jika Anda menentukan port yang berbeda, dan Anda menggunakan komponen jembatan MQTT untuk menyampaikan pesan MQTT ke broker lain, Anda harus

menggunakan MQTT bridge v2.1.0 atau yang lebih baru. Konfigurasikan untuk menggunakan port tempat broker MQTT beroperasi.
Jika Anda menentukan port yang berbeda, dan Anda menggunakan komponen detektor IP untuk mengelola titik akhir broker MQTT, Anda harus menggunakan detektor IP v2.1.0 atau yang lebih baru. Konfigurasikan untuk melaporkan port tempat broker MQTT beroperasi.

Default: `8883`

`listener.ssl.external.max_connections`

(Opsional) Jumlah maksimum koneksi bersamaan yang didukung oleh broker MQTT.

Default: `1024000`

`listener.ssl.external.max_conn_rate`

(Opsional) Jumlah maksimum koneksi baru per detik yang dapat diterima broker MQTT.

Default: `500`

`listener.ssl.external.rate_limit`

(Opsional) Batas bandwidth untuk semua koneksi ke broker MQTT. Tentukan bandwidth dan durasi untuk bandwidth yang dipisahkan oleh koma (`,`) dalam format berikut:`bandwidth,duration`. Misalnya, Anda dapat menentukan `50KB,5s` untuk membatasi broker MQTT hingga 50 kilobyte (KB) data setiap 5 detik.

`listener.ssl.external.handshake_timeout`

(Opsional) Jumlah waktu yang menunggu broker MQTT untuk menyelesaikan otentikasi koneksi baru.

Default: `15s`

`mqtt.max_packet_size`

(Opsional) Ukuran maksimum pesan MQTT.

Default: `268435455` (256 MB dikurangi 1)

`log.level`

(Opsional) Tingkat log untuk broker MQTT. Pilih dari salah satu pilihan berikut:

- `debug`
- `info`
- `notice`
- `warning`
- `error`
- `critical`
- `alert`
- `emergency`

Level log default adalah`warning`.

`requiresPrivilege`

(Opsional) Pada perangkat inti Linux, Anda dapat menentukan untuk menjalankan broker EMQX MQTT tanpa hak root atau administrator. Jika Anda menyetel opsi ini`false`, pengguna sistem yang menjalankan komponen ini harus menjadi anggota `docker` grup.

Default: `true`

`startupTimeoutSeconds`

(Opsional) Maksimum waktu dalam hitungan detik untuk memulai broker EMQX MQTT. Status komponen berubah menjadi `BROKEN` jika melebihi batas waktu ini.

Default: `90`

`ipcTimeoutSeconds`

(Opsional) Maksimum waktu dalam hitungan detik bagi komponen untuk menunggu inti Greengrass merespons permintaan komunikasi antarproses (IPC). Tingkatkan angka ini jika komponen ini melaporkan kesalahan batas waktu saat memeriksa apakah perangkat klien diotorisasi.

Default: `5`

`crtLogLevel`

(Opsional) Tingkat log untuk pustaka AWS Common Runtime (CRT).

Default ke level log broker EMQX MQTT (in). `log.level` `emqx`

`restartIdentifier`

(Opsional) Konfigurasikan opsi ini untuk memulai ulang broker EMQX MQTT. Ketika nilai konfigurasi ini berubah, komponen ini memulai ulang broker MQTT. Anda dapat menggunakan opsi ini untuk memaksa perangkat klien memutuskan sambungan.

`dockerOptions`

(Opsional) Konfigurasikan opsi ini hanya pada sistem operasi Linux untuk menambahkan parameter ke baris perintah Docker. Misalnya, untuk memetakan port tambahan, gunakan parameter `-p` Docker:

```
"-p 1883:1883"
```

`mergeConfigurationFiles`

(Opsional) Konfigurasikan opsi ini untuk menambah atau mengganti default dalam file konfigurasi EMQX yang ditentukan. Untuk informasi tentang file konfigurasi dan formatnya, lihat Konfigurasi dalam Dokumentasi EMQX 4.0. Nilai yang Anda tentukan ditambahkan ke file konfigurasi.

Contoh berikut memperbarui `etc/emqx.conf` file.

```
"mergeConfigurationFiles": {
    "etc/emqx.conf": "broker.sys_interval=30s\nbroker.sys_heartbeat=10s"
},
```

Selain file konfigurasi yang didukung oleh EMQX, Greengrass mendukung file yang mengkonfigurasi plugin autentikasi Greengrass untuk EMQX yang disebut. `etc/plugins/aws_greengrass_emqx_auth.conf` Ada dua opsi yang didukung, `auth_mode` dan`use_greengrass_managed_certificates`. Untuk menggunakan penyedia autentikasi lain, setel `auth_mode` opsi ke salah satu dari berikut ini:

- `enabled`— (Default) Gunakan penyedia otentikasi dan otorisasi Greengrass.
- `bypass_on_failure`— Gunakan penyedia otentikasi Greengrass, lalu gunakan penyedia otentikasi yang tersisa di rantai penyedia EMQX jika Greengrass menolak otentikasi atau otorisasi.
- `bypass`— Penyedia Greengrass dinonaktifkan. Otentikasi dan otorisasi kemudian ditangani oleh rantai penyedia EMQX.

Jika yatrue, opsi ini menunjukkan bahwa Greengrass mengelola sertifikat TLS broker. `use_greengrass_managed_certificates` Jika`false`, ini menunjukkan bahwa Anda memberikan sertifikat melalui sumber lain.

Contoh berikut memperbarui default dalam file konfigurasi. `etc/plugins/aws_greengrass_emqx_auth.conf`

```
"mergeConfigurationFiles": {
    "etc/plugins/aws_greengrass_emqx_auth.conf": "auth_mode=enabled\n
 use_greengrass_managed_certificates=true\n"
  },
```

> **Note**
>
> `aws.greengrass.clientdevices.mqtt.EMQX`memungkinkan Anda mengonfigurasi opsi yang sensitif terhadap keamanan. Ini termasuk pengaturan TLS, otentikasi, dan penyedia otorisasi. Konfigurasi yang disarankan adalah konfigurasi default yang menggunakan otentikasi TLS timbal balik dan penyedia Auth Perangkat Klien Greengrass.

`replaceConfigurationFiles`

(Opsional) Konfigurasikan opsi ini untuk mengganti file konfigurasi EMQX yang ditentukan. Nilai yang Anda tentukan menggantikan seluruh file konfigurasi yang ada. Anda tidak dapat menentukan `etc/emqx.conf` file di bagian ini. Anda harus menggunakan `mergeConfigurationFile` untuk memodifikasi`etc/emqx.conf`.

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk mengoperasikan broker MQTT pada port 443.

```
{
  "emqx": {
    "listener.ssl.external": "443",
    "listener.ssl.external.max_connections": "1024000",
    "listener.ssl.external.max_conn_rate": "500",
    "listener.ssl.external.rate_limit": "50KB,5s",
    "listener.ssl.external.handshake_timeout": "15s",
    "log.level": "warning"
```

```
  },
  "requiresPrivilege": "true",
  "startupTimeoutSeconds": "90",
  "ipcTimeoutSeconds": "5"
}
```

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.greengrass.clientdevices.mqtt.EMQX.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.clientdevices.mqtt.EMQX.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.greengrass.clientdevices.mqtt.EMQX.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.clientdevices.mqtt.EMQX.log -
  Tail 10 -Wait
  ```

Lisensi

Pada sistem operasi Windows, perangkat lunak ini menyertakan kode yang didistribusikan di bawah Persyaratan Lisensi Perangkat Lunak Microsoft - Komunitas Microsoft Visual Studio 2022. Dengan mengunduh perangkat lunak ini, Anda menyetujui ketentuan lisensi kode tersebut.

Komponen ini dirilis menurut Perjanjian Lisensi Perangkat Lunak Greengrass Core.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

v2.x

Versi	Perubahan
2.0.1	Versi diperbarui untuk perangkat klien autentikasi versi 2.5.0 rilis.
2.0.0	Versi broker MQTT 5 (EMQX) ini mengharapkan parameter konfigurasi yang berbeda dari versi 1.x. Jika Anda menggunakan konfigurasi non-default untuk versi 1.x, Anda harus memperbarui konfigurasi komponen untuk 2.x. Untuk informasi selengkapnya, lihat Konfigurasi. Fitur baru • Tingkatkan broker MQTT ke EMQX 5.1.1. • Mengaktifkan perubahan konfigurasi broker tanpa memulai ulang komponen. Pembaruan • Menambahkan bidang `emqxConfig` konfigurasi baru yang menggantikan bidang`emqx`,`mergeConfigurationFiles` , dan `replaceConfigurationFiles` konfigurasi.

v1.x

Versi	Perubahan
1.2.3	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana klien tidak dapat berinteraksi dengan EMQX setelah sebelumnya mengautentikasi dengan memutuskan dan mengautentikasi ulang klien.
1.2.2	Versi diperbarui untuk perangkat klien autentikasi versi 2.4.0 rilis.

Versi	Perubahan
1.2.1	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana komponen tidak akan memulai pada Windows jika Visual C++ Redistributable belum ada. • Pembaruan EMQX ke versi 4.4.14.
1.2.0	Menambahkan dukungan untuk rantai sertifikat.
1.1.0	Fitur baru • Menambahkan dukungan untuk konfigurasi EMQX termasuk opsi broker dan plug-in. Perbaikan bug dan peningkatan • Pembaruan EMQX ke versi 4.4.9.
1.0.1	Memperbaiki masalah selama jabat tangan TLS yang mengakibatkan beberapa klien MQTT gagal terhubung.
1.0.0	Versi awal.

Pemancar telemetri nukleus

Komponen pemancar telemetri nukleus (`aws.greengrass.telemetry.NucleusEmitter`) mengumpulkan data telemetri kesehatan sistem dan menerbitkannya terus menerus ke topik lokal dan topik MQTT. AWS IoT Core Komponen ini memungkinkan Anda untuk mengumpulkan telemetri sistem real-time pada perangkat inti Greengrass Anda. Untuk informasi tentang agen telemetri Greengrass yang menerbitkan data telemetri sistem ke Amazon, lihat. EventBridge Kumpulkan data telemetri kondisi sistem dari perangkat inti AWS IoT Greengrass

Secara default, komponen pemancar telemetri nukleus menerbitkan data telemetri setiap 60 detik ke topik penerbitan/langganan lokal berikut.

```
$local/greengrass/telemetry
```

Komponen pemancar telemetri nukleus tidak dipublikasikan ke topik AWS IoT Core MQTT secara default. Anda dapat mengonfigurasi komponen ini untuk dipublikasikan ke topik AWS IoT Core MQTT

saat Anda menerapkannya. Penggunaan topik MQTT untuk mempublikasikan data ke tunduk pada AWS Cloud harga. AWS IoT Core

AWS IoT Greengrassmenyediakan beberapa komponen komunitas untuk membantu Anda menganalisis dan memvisualisasikan data telemetri secara lokal di perangkat inti Anda menggunakan InfluxDB dan Grafana. Komponen-komponen ini menggunakan data telemetri dari komponen pemancar nukleus. Untuk informasi selengkapnya, lihat README untuk komponen penerbit InfluxDB.

Topik

Versi

Komponen ini memiliki versi berikut:

- 1.0.x

Tipe

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Dependensi

Saat Anda men-deploy komponen, AWS IoT Greengrass juga men-deploy versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

1.0.8

Tabel berikut mencantumkan dependensi untuk versi 1.0.8 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.4.0 <2.13.0	Keras

1.0.7

Tabel berikut mencantumkan dependensi untuk versi 1.0.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.4.0 <2.12.0	Keras

1.0.6

Tabel berikut mencantumkan dependensi untuk versi 1.0.6 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.4.0 <2.11.0	Keras

1.0.5

Tabel berikut mencantumkan dependensi untuk versi 1.0.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.4.0 <2.10.0	Keras

1.0.4

Tabel berikut mencantumkan dependensi untuk versi 1.0.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.4.0 <2.9.0	Keras

1.0.3

Tabel berikut mencantumkan dependensi untuk versi 1.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.4.0 <2.8.0	Keras

1.0.2

Tabel berikut mencantumkan dependensi untuk versi 1.0.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.4.0 <2.7.0	Keras

1.0.1

Tabel berikut mencantumkan dependensi untuk versi 1.0.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.4.0 <2.6.0	Keras

1.0.0

Tabel berikut mencantumkan dependensi untuk versi 1.0.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.4.0 <2.5.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`pubSubPublish`

(Opsional) Mendefinisikan apakah akan mempublikasikan data telemetri ke topik. `$local/greengrass/telemetry` Nilai yang didukung adalah `true` dan `false`.

Default: `true`

`mqttTopic`

(Opsional) Topik AWS IoT Core MQTT tempat komponen ini menerbitkan data telemetri.

Tetapkan nilai ini ke topik AWS IoT Core MQTT yang ingin Anda publikasikan data telemetri. Ketika nilai ini kosong, pemancar nukleus tidak mempublikasikan data telemetri ke. AWS Cloud

> **Note**
>
> Penggunaan topik MQTT untuk mempublikasikan data ke tunduk pada AWS Cloud harga. AWS IoT Core

Default: `""`

`telemetryPublishIntervalMs`

(Opsional) Jumlah waktu (dalam milidetik) di mana komponen menerbitkan data telemetri. Jika Anda menetapkan nilai ini lebih rendah dari nilai minimum yang didukung, komponen menggunakan nilai minimum sebagai gantinya.

> **Note**
>
> Interval publikasi yang lebih rendah menghasilkan penggunaan CPU yang lebih tinggi pada perangkat inti Anda. Kami menyarankan Anda memulai dengan interval publikasi default dan menyesuaikannya berdasarkan penggunaan CPU perangkat Anda.

Minimal: `500`

Default: `60000`

Example Contoh: Pembaruan gabungan konfigurasi

Contoh berikut menunjukkan pemutakhiran gabungan konfigurasi sampel yang memungkinkan penerbitan data telemetri setiap 5 detik ke `$local/greengrass/telemetry` topik dan topik MQTT. `greengrass/myTelemetry` AWS IoT Core

```
{
  "pubSubPublish": "true",
  "mqttTopic": "greengrass/myTelemetry",
  "telemetryPublishIntervalMs": 5000
}
```

Data output

Komponen ini menerbitkan metrik telemetri sebagai array JSON pada topik berikut.

Topik lokal: `$local/greengrass/telemetry`

Anda dapat memilih untuk juga mempublikasikan metrik telemetri ke topik MQTT. AWS IoT Core Untuk informasi selengkapnya tentang topik, lihat topik MQTT di Panduan Pengembang. AWS IoT Core

Example Contoh data


```
[
  {
    "A": "Average",
    "N": "CpuUsage",
    "NS": "SystemMetrics",
    "TS": 1627597331445,
    "U": "Percent",
    "V": 26.21981271562346
  },
  {
    "A": "Count",
    "N": "TotalNumberOfFDs",
    "NS": "SystemMetrics",
    "TS": 1627597331445,
    "U": "Count",
    "V": 7316
  },
  {
    "A": "Count",
    "N": "SystemMemUsage",
    "NS": "SystemMetrics",
    "TS": 1627597331445,
    "U": "Megabytes",
    "V": 10098
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsStarting",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
```

```
    "N": "NumberOfComponentsInstalled",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsStateless",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsStopping",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsBroken",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsRunning",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 7
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsErrored",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
```

```
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsNew",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsFinished",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 2
  }
]
```

Array keluaran berisi daftar metrik yang memiliki properti berikut:

`A`

Jenis agregasi untuk metrik.

Untuk `CpuUsage` metrik, properti ini disetel ke `Average` karena nilai metrik yang dipublikasikan adalah jumlah penggunaan CPU rata-rata sejak peristiwa publikasi terakhir.

Untuk semua metrik lainnya, pemancar nukleus tidak menggabungkan nilai metrik, dan properti ini disetel ke. `Count`

`N`

Nama metrik.

`NS`

Namespace metrik.

`TS`

Stempel waktu kapan data dikumpulkan.

U

Unit nilai metrik.

V

Nilai metrik.

Pemancar nukleus menerbitkan metrik berikut:

Nama	Penjelasan	
Sistem		
`SystemMemUsage`	Jumlah memori yang saat ini digunakan oleh semua aplikasi pada perangkat inti Greengrass, termasuk sistem operasi.	
`CpuUsage`	Jumlah CPU yang saat ini digunakan oleh semua aplikasi pada perangkat inti Greengrass, termasuk sistem operasi.	
`TotalNumberOfFDs`	Bilangan deskriptor file yang disimpan oleh sistem operasi perangkat inti Greengrass. Satu file deskriptor secara unik mengidentifikasi satu file yang terbuka.	
Inti Greengrass		
`NumberOfComponents Running`	Jumlah komponen yang berjalan pada perangkat inti Greengrass.	
`NumberOfComponents Errored`	Jumlah komponen yang berada dalam keadaan	

Nama	Penjelasan	
	kesalahan pada perangkat inti Greengrass.	
`NumberOfComponents Installed`	Jumlah komponen yang diinstal pada perangkat inti Greengrass.	
`NumberOfComponents Starting`	Jumlah komponen yang dimulai pada perangkat inti Greengrass.	
`NumberOfComponents New`	Jumlah komponen yang baru pada perangkat inti Greengrass.	
`NumberOfComponents Stopping`	Jumlah komponen yang berhenti pada perangkat inti Greengrass.	
`NumberOfComponents Finished`	Jumlah komponen yang diselesaikan pada perangkat inti Greengrass.	
`NumberOfComponents Broken`	Jumlah komponen yang rusak pada perangkat inti Greengrass.	
`NumberOfComponents Stateless`	Jumlah komponen yang stateless pada perangkat inti Greengrass.	

Penggunaan

Untuk menggunakan data telemetri kesehatan sistem, Anda dapat membuat komponen khusus yang berlangganan topik tempat pemancar inti menerbitkan data telemetri, dan bereaksi terhadap data tersebut sesuai kebutuhan. Karena komponen pemancar inti menyediakan opsi untuk mempublikasikan data telemetri ke topik lokal, Anda dapat berlangganan topik tersebut, dan

menggunakan data yang dipublikasikan untuk bertindak secara lokal di perangkat inti Anda. Perangkat inti kemudian dapat bereaksi terhadap data telemetri bahkan ketika konektivitas terbatas ke cloud.

Misalnya, Anda dapat mengonfigurasi komponen yang mendengarkan `$local/greengrass/telemetry` topik untuk data telemetri dan mengirim data ke komponen pengelola aliran untuk mengalirkan data Anda ke file. AWS Cloud Untuk informasi lebih lanjut tentang membuat komponen seperti itu, lihat Pesan lokal publikasi/berlangganan danBuat komponen kustom yang menggunakan stream manager.

File log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
1.0.8	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
1.0.7	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
1.0.6	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
1.0.5	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
1.0.4	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
1.0.3	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
1.0.2	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
1.0.1	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
1.0.0	Versi awal.

Penyedia PKCS #11

Komponen penyedia PKCS #11 (`aws.greengrass.crypto.Pkcs11Provider`) memungkinkan Anda mengonfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan modul keamanan perangkat keras (HSM) melalui antarmuka PKCS #11. Komponen ini memungkinkan Anda menyimpan file sertifikat dan kunci pribadi dengan aman sehingga tidak terekspos atau digandakan dalam perangkat lunak. Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras.

Untuk menyediakan perangkat inti Greengrass yang menyimpan sertifikat dan kunci privatnya di HSM, Anda harus menentukan komponen ini sebagai plugin penyediaan saat Anda menginstal perangkat lunak Core. AWS IoT Greengrass Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya manual.

AWS IoT Greengrassmenyediakan komponen ini sebagai file JAR yang dapat Anda unduh untuk ditentukan sebagai plugin penyediaan selama instalasi. Anda dapat mengunduh versi terbaru

dari file JAR komponen sebagai URL berikut: https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/aws.greengrass.crypto.Pkcs11Provider-latest.jar.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.0.x

Tipe

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Modul keamanan perangkat keras yang mendukung skema tanda tangan PKCS #1 v1.5 dan kunci RSA dengan ukuran kunci RSA-2048 (atau lebih besar) atau kunci ECC.

  > **Note**
  >
  > Untuk menggunakan modul keamanan perangkat keras dengan kunci ECC, Anda harus menggunakan Greengrass nucleus v2.5.6 atau yang lebih baru.
  > Untuk menggunakan modul keamanan perangkat keras dan manajer rahasia, Anda harus menggunakan modul keamanan perangkat keras dengan kunci RSA.

- Pustaka penyedia PKCS #11 yang dapat dimuat oleh perangkat lunak AWS IoT Greengrass Core saat runtime (menggunakan libdl) untuk menjalankan fungsi PKCS #11. Pustaka penyedia PKCS #11 harus mengimplementasikan operasi API PKCS #11 berikut:
  - `C_Initialize`
  - `C_Finalize`
  - `C_GetSlotList`
  - `C_GetSlotInfo`
  - `C_GetTokenInfo`
  - `C_OpenSession`
  - `C_GetSessionInfo`
  - `C_CloseSession`
  - `C_Login`
  - `C_Logout`
  - `C_GetAttributeValue`
  - `C_FindObjectsInit`
  - `C_FindObjects`
  - `C_FindObjectsFinal`
  - `C_DecryptInit`
  - `C_Decrypt`
  - `C_DecryptUpdate`
  - `C_DecryptFinal`
  - `C_SignInit`
  - `C_Sign`

  - `C_SignUpdate`
  - `C_SignFinal`
  - `C_GetMechanismList`
  - `C_GetMechanismInfo`
  - `C_GetInfo`
  - `C_GetFunctionList`
- Modul perangkat keras harus dapat diatasi dengan label slot, sebagaimana ditentukan di dalam spesifikasi PKCS#11.
- Anda harus menyimpan kunci pribadi dan sertifikat di HSM di slot yang sama, dan mereka harus menggunakan label objek dan ID objek yang sama, jika HSM mendukung ID objek.
- Sertifikat dan kunci pribadi harus dapat diselesaikan dengan label objek.
- Kunci pribadi harus memiliki izin berikut:
  - `sign`
  - `decrypt`
- (Opsional) Untuk menggunakan komponen manajer rahasia, Anda harus menggunakan versi 2.1.0 atau yang lebih baru, dan kunci pribadi harus memiliki izin berikut:
  - `unwrap`
  - `wrap`
- (Opsional) Jika Anda menggunakan pustaka TPM2 dan menjalankan inti Greengrass sebagai layanan, Anda harus menyediakan variabel lingkungan dengan lokasi penyimpanan PKCS #11. Contoh berikut adalah file layanan systemd dengan variabel lingkungan yang diperlukan:

```
[Unit]
Description=Greengrass Core
After=network.target

[Service]
Type=simple
PIDFile=/var/run/greengrass.pid
Environment=TPM2_PKCS11_STORE=/path/to/store/directory
RemainAfterExit=no
Restart=on-failure
RestartSec=10
ExecStart=/bin/sh /greengrass/v2/alts/current/distro/bin/loader

[Install]
```

```
WantedBy=multi-user.target
```

Dependensi

Saat Anda men-deploy komponen, AWS IoT Greengrass juga men-deploy versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.0.7

Tabel berikut mencantumkan dependensi untuk versi 2.0.7 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.3 <2.13.0	Lunak

2.0.6

Tabel berikut mencantumkan dependensi untuk versi 2.0.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.3 <2.12.0	Lunak

2.0.5

Tabel berikut mencantumkan dependensi untuk versi 2.0.5 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.3 <2.11.0	Lunak

2.0.4

Tabel berikut mencantumkan dependensi untuk versi 2.0.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.3 <2.10.0	Lunak

2.0.3

Tabel berikut mencantumkan dependensi untuk versi 2.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.3 <2.9.0	Lunak

2.0.2

Tabel berikut mencantumkan dependensi untuk versi 2.0.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.3 <2.8.0	Lunak

2.0.1

Tabel berikut mencantumkan dependensi untuk versi 2.0.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.3 <2.7.0	Lunak

2.0.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.3 <2.6.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`name`

Nama untuk konfigurasi PKCS #11.

`library`

Jalur file absolut ke pustaka implementasi PKCS #11 yang dapat dimuat oleh perangkat lunak AWS IoT Greengrass Core dengan libdl.

`slot`

ID slot yang berisi kunci pribadi dan sertifikat perangkat. Nilai ini berbeda dari indeks slot atau label slot.

`userPin`

PIN pengguna yang digunakan untuk mengakses slot.

Example Contoh: Pembaruan gabungan konfigurasi

```
{
  "name": "softhsm_pkcs11",
  "library": "/usr/lib/softhsm/libsofthsm2.so",
  "slot": 1,
  "userPin": "1234"
}
```

File log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.0.7	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.0.6	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.0.5	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.0.4	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.0.3	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.

Versi	Perubahan
2.0.2	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.0.1	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.0.0	Versi awal.

Secrets manager

Komponen secret manager (`aws.greengrass.SecretManager`) men-deploy rahasia dari AWS Secrets Manager ke perangkat inti Greengrass. Gunakan komponen ini untuk secara aman menggunakan kredensial, seperti kata sandi, dalam komponen kustom pada perangkat inti Greengrass Anda. Untuk informasi lebih lanjut tentang Secrets Manager, lihat Apa Itu AWS Secrets Manager? di Panduan Pengguna AWS Secrets Manager .

Untuk mengakses rahasia komponen ini di komponen Greengrass kustom Anda, gunakan GetSecret operasi Value di. AWS IoT Device SDK Untuk informasi selengkapnya, lihat Gunakan AWS IoT Device SDK untuk berkomunikasi dengan inti Greengrass, komponen lain, dan AWS IoT Core dan Ambil nilai-nilai rahasia.

Komponen ini mengenkripsi rahasia pada perangkat inti untuk menjaga kredensial dan kata sandi Anda tetap aman sampai Anda perlu menggunakannya. Ini menggunakan kunci pribadi perangkat inti untuk mengenkripsi dan mendekripsi rahasia.

Topik

- Versi
- Tipe
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- Berkas log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Peran perangkat Greengrass harus mengizinkan tindakan `secretsmanager:GetSecretValue`, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
```

```
      "Resource": [
        "arn:aws:secretsmanager:region:123456789012:secret:MySecret"
      ]
    }
  ]
}
```

> **Note**
>
> Jika Anda menggunakan AWS Key Management Service kunci yang dikelola pelanggan untuk mengenkripsi rahasia, peran perangkat juga harus mengizinkan tindakan. `kms:Decrypt`

Untuk informasi lebih lanjut tentang kebijakan IAM untuk Secrets Manager, lihat hal berikut di Panduan Pengguna AWS Secrets Manager :

- Otentikasi dan kontrol akses untuk AWS Secrets Manager
- Tindakan, sumber daya, dan kunci konteks yang dapat Anda gunakan dalam kebijakan IAM atau kebijakan rahasia untuk AWS Secrets Manager

- Komponen kustom harus menentukan kebijakan otorisasi yang memungkinkan `aws.greengrass#GetSecretValue` untuk mendapatkan rahasia yang Anda simpan dengan komponen ini. Dalam kebijakan otorisasi ini, Anda dapat membatasi akses komponen ke rahasia tertentu. Untuk informasi selengkapnya, lihat otorisasi IPC secret manager.
- (Opsional) Jika Anda menyimpan kunci pribadi dan sertifikat perangkat inti dalam modul keamanan perangkat keras (HSM), HSM harus mendukung kunci RSA, kunci pribadi harus memiliki `unwrap` izin, dan kunci publik harus memiliki izin. `wrap`

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Wajib	Deskripsi
`secretsmanager.` *region* `.amazonaws.com`	443	Ya	Unduh rahasia ke

Titik Akhir	Port	Wajib	Deskripsi
			perangkat inti.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.7 – 2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dan 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.13.0	Lunak

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.12.0	Lunak

2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.11.0	Lunak

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.10.0	Lunak

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.9.0	Lunak

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.8.0	Lunak

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.7.0	Lunak

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.6.0	Lunak

2.0.9

Tabel berikut mencantumkan dependensi untuk versi 2.0.9 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak

2.0.8

Tabel berikut mencantumkan dependensi untuk versi 2.0.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak

2.0.7

Tabel berikut mencantumkan dependensi untuk versi 2.0.7 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak

2.0.6

Tabel berikut mencantumkan dependensi untuk versi 2.0.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak

2.0.4 and 2.0.5

Tabel berikut mencantumkan dependensi untuk versi 2.0.4 dan 2.0.5 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`cloudSecrets`

Daftar rahasia Secrets Manager yang akan di-deploy ke perangkat inti. Anda dapat menentukan label untuk menentukan versi mana dari setiap rahasia yang akan di-deploy. Jika Anda tidak menentukan versi, komponen ini akan men-deploy versi dengan label penahapan `AWSCURRENT` terlampir. Untuk informasi selengkapnya, lihat Label penahapan di Panduan Pengguna AWS Secrets Manager .

Komponen manajer rahasia menyimpan rahasia secara lokal. Jika nilai rahasia berubah di Secrets Manager, komponen ini tidak secara otomatis mengambil nilai baru. Untuk memperbarui salinan lokal, berikan label baru pada rahasia dan konfigurasikan komponen ini untuk mengambil rahasia yang diidentifikasi oleh label baru.

Setiap objek berisi informasi berikut.

`arn`

ARN rahasia yang akan di-deploy. ARN rahasia dapat berupa ARN penuh atau ARN parsi. Kami menyarankan Anda menentukan ARN lengkap daripada ARN sebagian. Untuk informasi lebih lanjut, lihat Menemukan rahasia dari ARN sebagian. Berikut ini adalah contoh ARN lengkap dan ARN sebagian:

- ARN penuh: `arn:aws:secretsmanager:us-east-2:111122223333:secret:`*`SecretName`*`-abcdef`
- ARN sebagian: `arn:aws:secretsmanager:us-east-2:111122223333:secret:`*`SecretName`*

`labels`

(Opsional) Daftar label untuk mengidentifikasi versi rahasia yang akan di-deploy ke perangkat inti.

Setiap label harus berupa string.

Example Contoh: Pembaruan gabungan konfigurasi

```
{
  "cloudSecrets": [
    {
      "arn": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyGreengrassSecret-
abcdef"
    }
  ]
}
```

Berkas log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.8	Perbaikan bug dan peningkatan Memperbaiki masalah di mana manajer rahasia tidak menerima sebagian arn.
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.5	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.4	Perbaikan bug dan peningkatan Memperbaiki masalah saat rahasia yang di-cache dihapus saat manajer rahasia dikerahkan dan inti Greengrass dimulai ulang. Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.2	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.1	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.0	Fitur baru • Menambahkan dukungan untuk integrasi keamanan perangkat keras. Komponen manajer rahasia dapat mengenkripsi dan mendekripsi rahasia menggunakan kunci pribadi yang Anda simpan dalam modul

Versi	Perubahan
	keamanan perangkat keras (HSM). Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras. Perbaikan bug dan peningkatan • Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.9	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.8	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.7	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.0.6	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.
2.0.5	Perbaikan • Tambahkan dukungan untuk Wilayah dan AWS GovCloud (US) Wilayah AWS China.
2.0.4	Versi awal.

Tunneling yang aman

Dengan `aws.greengrass.SecureTunneling` komponen ini, Anda dapat membangun komunikasi dua arah yang aman dengan perangkat inti Greengrass yang terletak di belakang firewall terbatas.

Misalnya, bayangkan Anda memiliki perangkat inti Greengrass di belakang firewall yang melarang semua koneksi masuk. Tunneling aman menggunakan MQTT untuk mentransfer token akses ke perangkat dan kemudian menggunakan untuk membuat koneksi SSH WebSockets ke perangkat melalui firewall. Dengan terowongan AWS IoT terkelola ini, Anda dapat membuka koneksi SSH yang diperlukan untuk perangkat Anda. Untuk informasi selengkapnya tentang penggunaan tunneling AWS IoT aman untuk terhubung ke perangkat jarak jauh, lihat tunneling AWS IoT aman di Panduan Pengembang.AWS IoT

Komponen ini berlangganan broker pesan AWS IoT Core MQTT tentang `$aws/things/greengrass-core-device/tunnels/notify` topik tersebut untuk menerima pemberitahuan tunneling yang aman.

Topik

Versi

Komponen ini memiliki versi berikut:

- 1.0.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Arsitektur:

- Armv71
- Armv8 (AArch64)
- x86_64

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Minimal 32 MB ruang disk tersedia untuk komponen tunneling aman. Persyaratan ini tidak termasuk perangkat lunak inti Greengrass atau komponen lain yang berjalan pada perangkat yang sama.
- Minimal 16 MB RAM tersedia untuk komponen tunneling aman. Persyaratan ini tidak termasuk perangkat lunak inti Greengrass atau komponen lain yang berjalan pada perangkat yang sama. Untuk informasi selengkapnya, lihat Kontrol alokasi memori dengan opsi JVM.
- GNU C Library (glibc) versi 2.25 atau lebih tinggi dengan kernel Linux 3.2 atau lebih tinggi diperlukan untuk komponen tunneling aman versi 1.0.12 dan lebih besar. Versi sistem operasi dan pustaka yang melewati tanggal akhir masa pakai dukungan jangka panjangnya tidak didukung. Anda harus menggunakan sistem operasi dan perpustakaan dengan dukungan jangka panjang.
- Baik sistem operasi dan runtime Java harus diinstal sebagai 64 bit.
- Python 3.5 atau yang lebih baru diinstal pada perangkat inti Greengrass dan ditambahkan ke variabel lingkungan PATH.
- `libcrypto.so.1.1` diinstal pada perangkat inti Greengrass dan ditambahkan ke variabel lingkungan PATH.
- Buka lalu lintas keluar pada port 443 pada perangkat inti Greengrass.
- Nyalakan dukungan untuk layanan komunikasi yang ingin Anda gunakan untuk berkomunikasi dengan perangkat inti Greengrass. Misalnya, untuk membuka koneksi SSH ke perangkat, Anda harus mengaktifkan SSH pada perangkat itu.

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Wajib	Deskripsi
`data.tunneling.iot.`*`region`*`.amazonaws.com`	443	Ya	Membangun terowongan yang aman.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

1.0.19

Tabel berikut mencantumkan dependensi untuk versi 1.0.19 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <3.0.0	Lunak

1.0.18

Tabel berikut mencantumkan dependensi untuk versi 1.0.18 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak

1.0.16 – 1.0.17

Tabel berikut mencantumkan dependensi untuk versi 1.0.16 hingga 1.0.17 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak

1.0.14 – 1.0.15

Tabel berikut mencantumkan dependensi untuk versi 1.0.14 hingga 1.0.15 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak

1.0.11 – 1.0.13

Tabel berikut mencantumkan dependensi untuk versi 1.0.11 — 1.0.13 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak

1.0.10

Tabel berikut mencantumkan dependensi untuk versi 1.0.10 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak

1.0.9

Tabel berikut mencantumkan dependensi untuk versi 1.0.9 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak

1.0.8

Tabel berikut mencantumkan dependensi untuk versi 1.0.8 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak

1.0.5 - 1.0.7

Tabel berikut mencantumkan dependensi untuk versi 1.0.5 hingga 1.0.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak

1.0.4

Tabel berikut mencantumkan dependensi untuk versi 1.0.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak

1.0.3

Tabel berikut mencantumkan dependensi untuk versi 1.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak

1.0.2

Tabel berikut mencantumkan dependensi untuk versi 1.0.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak

1.0.1

Tabel berikut mencantumkan dependensi untuk versi 1.0.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak

1.0.0

Tabel berikut mencantumkan dependensi untuk versi 1.0.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`OS_DIST_INFO`

(Opsional) Sistem operasi perangkat inti Anda. Secara default, komponen mencoba mengidentifikasi secara otomatis sistem operasi yang berjalan pada perangkat inti Anda. Jika komponen gagal memulai dengan nilai default, gunakan nilai ini untuk menentukan sistem operasi. Untuk daftar sistem operasi yang didukung untuk komponen ini, lihat Persyaratan perangkat.

Nilai ini dapat berupa salah satu dari yang berikut: `auto`, `ubuntu`, `amzn2`, `raspberrypi`.

Default: `auto`

`accessControl`

(Opsional) Objek yang berisi kebijakan otorisasi yang memungkinkan komponen untuk berlangganan topik pemberitahuan tunneling aman.

> **Note**
>
> Jangan mengubah parameter konfigurasi ini jika deployment Anda menargetkan grup objek. Jika penerapan Anda menargetkan perangkat inti individual, dan Anda ingin

membatasi langganannya ke topik perangkat, tentukan nama perangkat inti. Dalam `resources` nilai dalam kebijakan otorisasi perangkat, ganti wildcard topik MQTT dengan nama benda perangkat.

```
{
  "aws.greengrass.ipc.mqttproxy": {
    "aws.iot.SecureTunneling:mqttproxy:1": {
      "policyDescription": "Access to tunnel notification pubsub topic",
      "operations": [
        "aws.greengrass#SubscribeToIoTCore"
      ],
      "resources": [
        "$aws/things/+/tunnels/notify"
      ]
    }
  }
}
```

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk memungkinkan komponen ini untuk membuka terowongan aman pada perangkat inti bernama **`MyGreengrassCore`** yang menjalankan Ubuntu.

```
{
  "OS_DIST_INFO": "ubuntu",
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "aws.iot.SecureTunneling:mqttproxy:1": {
        "policyDescription": "Access to tunnel notification pubsub topic",
        "operations": [
          "aws.greengrass#SubscribeToIoTCore"
        ],
        "resources": [
          "$aws/things/MyGreengrassCore/tunnels/notify"
        ]
      }
    }
  }
}
```

File log lokal

Komponen ini menggunakan file log berikut.

```
/greengrass/v2/logs/aws.greengrass.SecureTunneling.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* dengan jalur ke folder AWS IoT Greengrass root.

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

Lisensi

Komponen ini mencakup perangkat lunak/lisensi pihak ketiga berikut:

- AWS IoT Klien Perangkat/Lisensi Apache 2.0
- AWS IoT Device SDK for Java/Apache License 2.0
- gson/Apache License 2.0
- log4j/Apache License 2.0
- slf4j/Apache License 2.0

Penggunaan

Untuk menggunakan komponen tunneling aman di perangkat Anda, lakukan hal berikut:

1. Terapkan komponen tunneling aman ke perangkat Anda.
2. Buka konsol AWS IoT. Dari menu sebelah kiri, pilih Tindakan jarak jauh, lalu pilih Terowongan aman.
3. Buat terowongan ke perangkat Greengrass Anda.
4. Unduh token akses sumber.
5. Gunakan proxy lokal dengan token akses sumber untuk terhubung ke tujuan Anda. Untuk informasi selengkapnya, lihat Cara menggunakan proxy lokal di Panduan AWS IoT Pengembang.

Lihat juga

- AWS IoT tunneling aman di Panduan Pengembang AWS IoT
- Cara menggunakan proxy lokal di Panduan AWS IoT Pengembang

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
1.0.19	Perbaikan bug dan peningkatan • Memutakhirkan AWS IoT Device Client yang mendasari yang dipanggil oleh komponen dari versi 1.8.0 ke versi 1.9.0. • Meningkatkan batas terowongan bersamaan menjadi 20 terowongan pada tingkat komponen. • Meningkatkan batas waktu AWS IoT Greengrass Core IPC default dari 3 detik menjadi 10 detik. **Warning** Jika Anda menggunakan proxy lokal tunneling aman sebagai klien sumber terowongan, jangan perbarui komponen Anda ke versi ini sampai Anda juga memutakhirkan proxy lokal ke versi 3.1.1 atau yang lebih baru.
1.0.18	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
1.0.17	Perbaikan bug dan peningkatan • Memperbaiki masalah pembersihan utas yang menghalangi pengguna membuat terowongan. Komponen ini sekarang akan membersihkan utas baik setelah menerima CloseTunnel sinyal atau jika terowongan kedaluwarsa setelah 12 jam.
1.0.16	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.

Versi	Perubahan
1.0.15	Perbaikan bug dan peningkatan • Memperbaiki masalah startup untuk pengguna yang tidak memiliki direktori home di perangkat. Komponen tunneling aman sekarang dimulai tanpa membuat direktori untuk dokumen bayangan.
1.0.14	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
1.0.13	Perbaikan bug dan peningkatan • Memperbaiki masalah saat proses klien yatim piatu mencegah lebih dari satu terowongan menargetkan perangkat.
1.0.12	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk x86_64 (AMD64) dan ARMv8 (Aarch64) saat berjalan di Raspberry Pi OS.
1.0.11	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
1.0.10	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
1.0.9	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
1.0.8	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
1.0.7	Perbaikan bug dan peningkatan • Memperbaiki masalah saat komponen terputus saat Anda mentransfer file besar melalui SCP.
1.0.6	Versi ini berisi perbaikan bug.
1.0.5	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
1.0.4	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
1.0.3	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
1.0.2	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
1.0.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.

Versi	Perubahan
1.0.0	Versi awal.

Pengelola bayangan

Komponen shadow manager (`aws.greengrass.ShadowManager`) memungkinkan layanan bayangan lokal pada perangkat inti Anda. Layanan bayangan lokal memungkinkan komponen untuk menggunakan komunikasi antar proses untuk berinteraksi dengan bayangan lokal. Komponen shadow manager mengelola penyimpanan dokumen bayangan lokal, dan juga menangani sinkronisasi status bayangan lokal dengan layanan AWS IoT Device Shadow.

Untuk informasi selengkapnya tentang bagaimana perangkat inti Greengrass dapat berinteraksi dengan bayangan, lihat. Berinteraksilah dengan bayangan perangkat

Topik

- Versi
- Tipe
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- Berkas log lokal
- Changelog

Versi

Komponen ini memiliki versi berikut:

- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen plugin (`aws.greengrass.plugin`). Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen ini di perangkat inti.

Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- (Opsional) Untuk menyinkronkan bayangan ke layanan AWS IoT Device Shadow, kebijakan perangkat inti Greengrass harus mengizinkan AWS IoT tindakan kebijakan bayangan berikut: AWS IoT Core
  - `iot:GetThingShadow`
  - `iot:UpdateThingShadow`
  - `iot:DeleteThingShadow`

  Untuk informasi selengkapnya tentang AWS IoT Core kebijakan ini, lihat tindakan AWS IoT Core kebijakan di Panduan AWS IoT Pengembang.

  Untuk informasi selengkapnya tentang AWS IoT kebijakan minimal, lihat Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2
- Komponen shadow manager didukung untuk berjalan di VPC.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.3.5 – 2.3.8

Tabel berikut mencantumkan dependensi untuk versi 2.3.5 hingga 2.3.8 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.13.0	Lunak

2.3.3 and 2.3.4

Tabel berikut mencantumkan dependensi untuk versi 2.3.3 dan 2.3.4 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.12.0	Lunak

2.3.2

Tabel berikut mencantumkan dependensi untuk versi 2.3.2 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.11.0	Lunak

2.3.0 and 2.3.1

Tabel berikut mencantumkan dependensi untuk versi 2.3.0 dan 2.3.1 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.5.0 <2.10.0	Lunak

2.2.3 and 2.2.4

Tabel berikut mencantumkan dependensi untuk versi 2.2.3 dan 2.2.4 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <3.0.0	Lunak

2.2.2

Tabel berikut mencantumkan dependensi untuk versi 2.2.2 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.9.0	Lunak

2.2.1

Tabel berikut mencantumkan dependensi untuk versi 2.2.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.8.0	Lunak

2.1.1 and 2.2.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 dan 2.2.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.7.0	Lunak

2.0.5 - 2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.5 hingga 2.1.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.6.0	Lunak

2.0.3 and 2.0.4

Tabel berikut mencantumkan dependensi untuk versi 2.0.3 dan 2.0.4 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.5.0	Lunak

2.0.1 and 2.0.2

Tabel berikut mencantumkan dependensi untuk versi 2.0.1 dan 2.0.2 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.4.0	Lunak

2.0.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.2.0 <2.3.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

2.3.x

`strategy`

(Opsional) Strategi yang digunakan komponen ini untuk menyinkronkan bayangan antara AWS IoT Core dan perangkat inti.

Objek ini berisi informasi berikut.

`type`

(Opsional) Jenis strategi yang digunakan komponen ini untuk menyinkronkan bayangan antara AWS IoT Core dan perangkat inti. Pilih dari salah satu pilihan berikut:

- `realTime`— Sinkronkan bayangan dengan AWS IoT Core setiap kali pembaruan bayangan terjadi.
- `periodic`— Sinkronkan bayangan dengan AWS IoT Core interval reguler yang Anda tentukan dengan parameter `delay` konfigurasi.

Default: `realTime`

`delay`

(Opsional) Interval dalam detik di mana komponen ini menyinkronkan bayangan dengan AWS IoT Core, saat Anda menentukan strategi `periodic` sinkronisasi.

> **Note**
>
> Parameter ini diperlukan jika Anda menentukan strategi `periodic` sinkronisasi.

`synchronize`

(Opsional) Pengaturan sinkronisasi yang menentukan bagaimana bayangan disinkronkan dengan AWS Cloud.

> **Note**
>
> Anda harus membuat pembaruan konfigurasi dengan properti ini untuk menyinkronkan bayangan dengan AWS Cloud.

Objek ini berisi informasi berikut.

`coreThing`

(Opsional) Bayangan perangkat inti untuk disinkronkan. Objek ini berisi informasi berikut.

`classic`

(Opsional) Secara default, pengelola bayangan menyinkronkan status lokal bayangan klasik untuk perangkat inti Anda dengan. AWS Cloud Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik, atur ini ke `false`.

Default: `true`

`namedShadows`

(Opsional) Daftar bayangan perangkat inti bernama untuk disinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

> **Warning**
>
> AWS IoT Greengrass Layanan ini menggunakan bayangan `AWSManagedGreengrassV2Deployment` bernama untuk mengelola penerapan yang menargetkan perangkat inti individual. Bayangan bernama ini dicadangkan untuk digunakan oleh AWS IoT Greengrass layanan. Jangan perbarui atau hapus bayangan bernama ini.

`shadowDocumentsMap`

(Opsional) Bayangan perangkat tambahan untuk disinkronkan. Menggunakan parameter konfigurasi ini memudahkan untuk menentukan dokumen bayangan. Kami menyarankan Anda menggunakan parameter ini alih-alih `shadowDocuments` objek.

> **Note**
>
> Jika Anda menentukan `shadowDocumentsMap` objek, Anda tidak harus menentukan `shadowDocuments` objek.

Setiap objek berisi informasi berikut.

`thingName`

Konfigurasi bayangan untuk *`ThingName`* untuk konfigurasi bayangan ini.

`classic`

(Opsional) Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik untuk `thingName` perangkat, setel ini ke`false`.

`namedShadows`

Daftar bayangan bernama yang ingin Anda sinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

`shadowDocuments`

(Opsional) Daftar bayangan perangkat tambahan untuk disinkronkan. Kami menyarankan Anda menggunakan `shadowDocumentsMap` parameter sebagai gantinya.

> **Note**
>
> Jika Anda menentukan `shadowDocuments` objek, Anda tidak harus menentukan `shadowDocumentsMap` objek.

Setiap objek dalam daftar ini berisi informasi berikut.

`thingName`

Nama objek perangkat untuk menyinkronkan bayangan.

`classic`

(Opsional) Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik untuk `thingName` perangkat, setel ini ke`false`.

Default: `true`

`namedShadows`

(Opsional) Daftar bayangan perangkat bernama yang ingin Anda sinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

`direction`

(Opsional) Arah untuk menyinkronkan bayangan antara layanan bayangan lokal dan AWS Cloud. Anda dapat mengonfigurasi opsi ini untuk mengurangi bandwidth dan koneksi ke file AWS Cloud. Pilih dari salah satu pilihan berikut:

- `betweenDeviceAndCloud`— Sinkronisasi bayangan antara layanan bayangan lokal dan. AWS Cloud
- `deviceToCloud`— Kirim pembaruan bayangan dari layanan bayangan lokal ke AWS Cloud, dan abaikan pembaruan bayangan dari AWS Cloud.
- `cloudToDevice`— Terima pembaruan bayangan dari AWS Cloud, dan jangan mengirim pembaruan bayangan dari layanan bayangan lokal ke AWS Cloud.

Default: `BETWEEN_DEVICE_AND_CLOUD`

`rateLimits`

(Opsional) Pengaturan yang menentukan batas tarif untuk permintaan layanan bayangan.

Objek ini berisi informasi berikut.

`maxOutboundSyncUpdatesPerSecond`

(Opsional) Jumlah maksimum permintaan sinkronisasi per detik yang ditransmisikan perangkat.

Default: 100 permintaan/detik

`maxTotalLocalRequestsRate`

(Opsional) Jumlah maksimum permintaan IPC lokal per detik yang dikirim ke perangkat inti.

Default: 200 permintaan/detik

`maxLocalRequestsPerSecondPerThing`

(Opsional) Jumlah maksimum permintaan IPC lokal per detik yang dikirim untuk setiap hal IoT yang terhubung.

Default: 20 permintaan/detik untuk setiap objek

> **Note**
>
> Parameter batas tingkat ini menentukan jumlah maksimum permintaan per detik untuk layanan bayangan lokal. Jumlah maksimum permintaan per detik untuk layanan AWS IoT Device Shadow tergantung pada Anda Wilayah AWS. Untuk informasi selengkapnya, lihat batasan untuk AWS IoT Device Shadow Service API di Referensi Umum Amazon Web.

`shadowDocumentSizeLimitBytes`

(Opsional) Ukuran maksimum yang diizinkan dari setiap dokumen status JSON untuk bayangan lokal.

Jika Anda meningkatkan nilai ini, Anda juga harus meningkatkan batas sumber daya untuk dokumen keadaan JSON untuk bayangan cloud. Untuk informasi selengkapnya, lihat batasan untuk AWS IoT Device Shadow Service API di Referensi Umum Amazon Web.

Default: 8192 byte

Maksimum: 30720 byte

Example Contoh: Pembaruan gabungan konfigurasi

Contoh berikut menunjukkan pembaruan gabungan konfigurasi sampel dengan semua parameter konfigurasi yang tersedia untuk komponen manajer bayangan.

```
{
   "strategy":{
      "type":"periodic",
      "delay":300
   },
   "synchronize":{
      "shadowDocumentsMap":{
         "MyDevice1":{
            "classic":false,
            "namedShadows":[
               "MyShadowA",
```

```
                "MyShadowB"
             ]
          },
          "MyDevice2":{
             "classic":true,
             "namedShadows":[]
          }
       },
       "direction":"betweenDeviceAndCloud"
    },
    "rateLimits":{
       "maxOutboundSyncUpdatesPerSecond":100,
       "maxTotalLocalRequestsRate":200,
       "maxLocalRequestsPerSecondPerThing":20
    },
    "shadowDocumentSizeLimitBytes":8192
}
```

2.2.x

`strategy`

(Opsional) Strategi yang digunakan komponen ini untuk menyinkronkan bayangan antara AWS IoT Core dan perangkat inti.

Objek ini berisi informasi berikut.

`type`

(Opsional) Jenis strategi yang digunakan komponen ini untuk menyinkronkan bayangan antara AWS IoT Core dan perangkat inti. Pilih dari salah satu pilihan berikut:

- `realTime`— Sinkronkan bayangan dengan AWS IoT Core setiap kali pembaruan bayangan terjadi.
- `periodic`— Sinkronkan bayangan dengan AWS IoT Core interval reguler yang Anda tentukan dengan parameter `delay` konfigurasi.

Default: `realTime`

`delay`

(Opsional) Interval dalam detik di mana komponen ini menyinkronkan bayangan dengan AWS IoT Core, saat Anda menentukan strategi `periodic` sinkronisasi.

> **Note**
>
> Parameter ini diperlukan jika Anda menentukan strategi `periodic` sinkronisasi.

`synchronize`

(Opsional) Pengaturan sinkronisasi yang menentukan bagaimana bayangan disinkronkan dengan AWS Cloud.

> **Note**
>
> Anda harus membuat pembaruan konfigurasi dengan properti ini untuk menyinkronkan bayangan dengan AWS Cloud.

Objek ini berisi informasi berikut.

`coreThing`

(Opsional) Bayangan perangkat inti untuk disinkronkan. Objek ini berisi informasi berikut.

`classic`

(Opsional) Secara default, pengelola bayangan menyinkronkan status lokal bayangan klasik untuk perangkat inti Anda dengan. AWS Cloud Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik, atur ini ke `false`.

Default: `true`

`namedShadows`

(Opsional) Daftar bayangan perangkat inti bernama untuk disinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

> **Warning**
>
> AWS IoT Greengrass Layanan ini menggunakan bayangan `AWSManagedGreengrassV2Deployment` bernama untuk mengelola penerapan yang menargetkan perangkat inti individual. Bayangan bernama ini dicadangkan untuk digunakan oleh AWS IoT Greengrass layanan. Jangan perbarui atau hapus bayangan bernama ini.

`shadowDocumentsMap`

(Opsional) Bayangan perangkat tambahan untuk disinkronkan. Menggunakan parameter konfigurasi ini memudahkan untuk menentukan dokumen bayangan. Kami menyarankan Anda menggunakan parameter ini alih-alih `shadowDocuments` objek.

> **Note**
>
> Jika Anda menentukan `shadowDocumentsMap` objek, Anda tidak harus menentukan `shadowDocuments` objek.

Setiap objek berisi informasi berikut.

`thingName`

Konfigurasi bayangan untuk *`ThingName`* untuk konfigurasi bayangan ini.

`classic`

(Opsional) Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik untuk `thingName` perangkat, setel ini ke`false`.

`namedShadows`

Daftar bayangan bernama yang ingin Anda sinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

`shadowDocuments`

(Opsional) Daftar bayangan perangkat tambahan untuk disinkronkan. Kami menyarankan Anda menggunakan `shadowDocumentsMap` parameter sebagai gantinya.

> **Note**
>
> Jika Anda menentukan `shadowDocuments` objek, Anda tidak harus menentukan `shadowDocumentsMap` objek.

Setiap objek dalam daftar ini berisi informasi berikut.

`thingName`

Nama objek perangkat untuk menyinkronkan bayangan.

`classic`

(Opsional) Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik untuk `thingName` perangkat, setel ini ke`false`.

Default: `true`

`namedShadows`

(Opsional) Daftar bayangan perangkat bernama yang ingin Anda sinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

`direction`

(Opsional) Arah untuk menyinkronkan bayangan antara layanan bayangan lokal dan AWS Cloud. Anda dapat mengonfigurasi opsi ini untuk mengurangi bandwidth dan koneksi ke file AWS Cloud. Pilih dari salah satu pilihan berikut:

- `betweenDeviceAndCloud`— Sinkronisasi bayangan antara layanan bayangan lokal dan. AWS Cloud
- `deviceToCloud`— Kirim pembaruan bayangan dari layanan bayangan lokal ke AWS Cloud, dan abaikan pembaruan bayangan dari AWS Cloud.
- `cloudToDevice`— Terima pembaruan bayangan dari AWS Cloud, dan jangan mengirim pembaruan bayangan dari layanan bayangan lokal ke AWS Cloud.

Default: `BETWEEN_DEVICE_AND_CLOUD`

`rateLimits`

(Opsional) Pengaturan yang menentukan batas tarif untuk permintaan layanan bayangan.

Objek ini berisi informasi berikut.

`maxOutboundSyncUpdatesPerSecond`

(Opsional) Jumlah maksimum permintaan sinkronisasi per detik yang ditransmisikan perangkat.

Default: 100 permintaan/detik

`maxTotalLocalRequestsRate`

(Opsional) Jumlah maksimum permintaan IPC lokal per detik yang dikirim ke perangkat inti.

Default: 200 permintaan/detik

`maxLocalRequestsPerSecondPerThing`

(Opsional) Jumlah maksimum permintaan IPC lokal per detik yang dikirim untuk setiap hal IoT yang terhubung.

Default: 20 permintaan/detik untuk setiap objek

> **Note**
>
> Parameter batas tingkat ini menentukan jumlah maksimum permintaan per detik untuk layanan bayangan lokal. Jumlah maksimum permintaan per detik untuk layanan AWS IoT Device Shadow tergantung pada Anda Wilayah AWS. Untuk informasi selengkapnya, lihat batasan untuk AWS IoT Device Shadow Service API di Referensi Umum Amazon Web.

`shadowDocumentSizeLimitBytes`

(Opsional) Ukuran maksimum yang diizinkan dari setiap dokumen status JSON untuk bayangan lokal.

Jika Anda meningkatkan nilai ini, Anda juga harus meningkatkan batas sumber daya untuk dokumen keadaan JSON untuk bayangan cloud. Untuk informasi selengkapnya, lihat batasan untuk AWS IoT Device Shadow Service API di Referensi Umum Amazon Web.

Default: 8192 byte

Maksimum: 30720 byte

Example Contoh: Pembaruan gabungan konfigurasi

Contoh berikut menunjukkan pembaruan gabungan konfigurasi sampel dengan semua parameter konfigurasi yang tersedia untuk komponen manajer bayangan.

```
{
   "strategy":{
      "type":"periodic",
      "delay":300
```

```
    },
    "synchronize":{
       "shadowDocumentsMap":{
          "MyDevice1":{
             "classic":false,
             "namedShadows":[
                "MyShadowA",
                "MyShadowB"
             ]
          },
          "MyDevice2":{
             "classic":true,
             "namedShadows":[]
          }
       },
       "direction":"betweenDeviceAndCloud"
    },
    "rateLimits":{
       "maxOutboundSyncUpdatesPerSecond":100,
       "maxTotalLocalRequestsRate":200,
       "maxLocalRequestsPerSecondPerThing":20
    },
    "shadowDocumentSizeLimitBytes":8192
}
```

2.1.x

`strategy`

(Opsional) Strategi yang digunakan komponen ini untuk menyinkronkan bayangan antara AWS IoT Core dan perangkat inti.

Objek ini berisi informasi berikut.

`type`

(Opsional) Jenis strategi yang digunakan komponen ini untuk menyinkronkan bayangan antara AWS IoT Core dan perangkat inti. Pilih dari salah satu pilihan berikut:

- `realTime`— Sinkronkan bayangan dengan AWS IoT Core setiap kali pembaruan bayangan terjadi.
- `periodic`— Sinkronkan bayangan dengan AWS IoT Core interval reguler yang Anda tentukan dengan parameter `delay` konfigurasi.

Default: `realTime`

`delay`

(Opsional) Interval dalam detik di mana komponen ini menyinkronkan bayangan dengan AWS IoT Core, saat Anda menentukan strategi `periodic` sinkronisasi.

> **Note**
>
> Parameter ini diperlukan jika Anda menentukan strategi `periodic` sinkronisasi.

`synchronize`

(Opsional) Pengaturan sinkronisasi yang menentukan bagaimana bayangan disinkronkan dengan AWS Cloud.

> **Note**
>
> Anda harus membuat pembaruan konfigurasi dengan properti ini untuk menyinkronkan bayangan dengan AWS Cloud.

Objek ini berisi informasi berikut.

`coreThing`

(Opsional) Bayangan perangkat inti untuk disinkronkan. Objek ini berisi informasi berikut.

`classic`

(Opsional) Secara default, pengelola bayangan menyinkronkan status lokal bayangan klasik untuk perangkat inti Anda dengan. AWS Cloud Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik, atur ini ke `false`.

Default: `true`

`namedShadows`

(Opsional) Daftar bayangan perangkat inti bernama untuk disinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

> **Warning**
>
> AWS IoT Greengrass Layanan ini menggunakan bayangan `AWSManagedGreengrassV2Deployment` bernama untuk mengelola penerapan yang menargetkan perangkat inti individual. Bayangan bernama ini dicadangkan untuk digunakan oleh AWS IoT Greengrass layanan. Jangan perbarui atau hapus bayangan bernama ini.

`shadowDocumentsMap`

(Opsional) Bayangan perangkat tambahan untuk disinkronkan. Menggunakan parameter konfigurasi ini memudahkan untuk menentukan dokumen bayangan. Kami menyarankan Anda menggunakan parameter ini alih-alih `shadowDocuments` objek.

> **Note**
>
> Jika Anda menentukan `shadowDocumentsMap` objek, Anda tidak harus menentukan `shadowDocuments` objek.

Setiap objek berisi informasi berikut.

`thingName`

Konfigurasi bayangan untuk *`ThingName`* untuk konfigurasi bayangan ini.

`classic`

(Opsional) Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik untuk `thingName` perangkat, setel ini ke`false`.

`namedShadows`

Daftar bayangan bernama yang ingin Anda sinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

`shadowDocuments`

(Opsional) Daftar bayangan perangkat tambahan untuk disinkronkan. Kami menyarankan Anda menggunakan `shadowDocumentsMap` parameter sebagai gantinya.

> **Note**
>
> Jika Anda menentukan `shadowDocuments` objek, Anda tidak harus menentukan `shadowDocumentsMap` objek.

Setiap objek dalam daftar ini berisi informasi berikut.

`thingName`

Nama objek perangkat untuk menyinkronkan bayangan.

`classic`

(Opsional) Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik untuk `thingName` perangkat, setel ini ke`false`.

Default: `true`

`namedShadows`

(Opsional) Daftar bayangan perangkat bernama yang ingin Anda sinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

`rateLimits`

(Opsional) Pengaturan yang menentukan batas tarif untuk permintaan layanan bayangan.

Objek ini berisi informasi berikut.

`maxOutboundSyncUpdatesPerSecond`

(Opsional) Jumlah maksimum permintaan sinkronisasi per detik yang ditransmisikan perangkat.

Default: 100 permintaan/detik

`maxTotalLocalRequestsRate`

(Opsional) Jumlah maksimum permintaan IPC lokal per detik yang dikirim ke perangkat inti.

Default: 200 permintaan/detik

`maxLocalRequestsPerSecondPerThing`

(Opsional) Jumlah maksimum permintaan IPC lokal per detik yang dikirim untuk setiap hal IoT yang terhubung.

Default: 20 permintaan/detik untuk setiap objek

> **Note**
>
> Parameter batas tingkat ini menentukan jumlah maksimum permintaan per detik untuk layanan bayangan lokal. Jumlah maksimum permintaan per detik untuk layanan AWS IoT Device Shadow tergantung pada Anda Wilayah AWS. Untuk informasi selengkapnya, lihat batasan untuk AWS IoT Device Shadow Service API di Referensi Umum Amazon Web.

`shadowDocumentSizeLimitBytes`

(Opsional) Ukuran maksimum yang diizinkan dari setiap dokumen status JSON untuk bayangan lokal.

Jika Anda meningkatkan nilai ini, Anda juga harus meningkatkan batas sumber daya untuk dokumen keadaan JSON untuk bayangan cloud. Untuk informasi selengkapnya, lihat batasan untuk AWS IoT Device Shadow Service API di Referensi Umum Amazon Web.

Default: 8192 byte

Maksimum: 30720 byte

Example Contoh: Pembaruan gabungan konfigurasi

Contoh berikut menunjukkan pembaruan gabungan konfigurasi sampel dengan semua parameter konfigurasi yang tersedia untuk komponen manajer bayangan.

```
{
   "strategy":{
      "type":"periodic",
      "delay":300
   },
   "synchronize":{
      "shadowDocumentsMap":{
         "MyDevice1":{
            "classic":false,
            "namedShadows":[
               "MyShadowA",
               "MyShadowB"
```

```
            ]
         },
         "MyDevice2":{
            "classic":true,
            "namedShadows":[]
         }
      },
      "direction":"betweenDeviceAndCloud"
   },
   "rateLimits":{
      "maxOutboundSyncUpdatesPerSecond":100,
      "maxTotalLocalRequestsRate":200,
      "maxLocalRequestsPerSecondPerThing":20
   },
   "shadowDocumentSizeLimitBytes":8192
}
```

2.0.x

`synchronize`

(Opsional) Pengaturan sinkronisasi yang menentukan bagaimana bayangan disinkronkan dengan AWS Cloud.

> **Note**
>
> Anda harus membuat pembaruan konfigurasi dengan properti ini untuk menyinkronkan bayangan dengan AWS Cloud.

Objek ini berisi informasi berikut.

`coreThing`

(Opsional) Bayangan perangkat inti untuk disinkronkan. Objek ini berisi informasi berikut.

`classic`

(Opsional) Secara default, pengelola bayangan menyinkronkan status lokal bayangan klasik untuk perangkat inti Anda dengan. AWS Cloud Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik, atur ini ke `false`.

Default: `true`

`namedShadows`

(Opsional) Daftar bayangan perangkat inti bernama untuk disinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

> **Warning**
>
> AWS IoT Greengrass Layanan ini menggunakan bayangan `AWSManagedGreengrassV2Deployment` bernama untuk mengelola penerapan yang menargetkan perangkat inti individual. Bayangan bernama ini dicadangkan untuk digunakan oleh AWS IoT Greengrass layanan. Jangan perbarui atau hapus bayangan bernama ini.

`shadowDocumentsMap`

(Opsional) Bayangan perangkat tambahan untuk disinkronkan. Menggunakan parameter konfigurasi ini memudahkan untuk menentukan dokumen bayangan. Kami menyarankan Anda menggunakan parameter ini alih-alih `shadowDocuments` objek.

> **Note**
>
> Jika Anda menentukan `shadowDocumentsMap` objek, Anda tidak harus menentukan `shadowDocuments` objek.

Setiap objek berisi informasi berikut.

`thingName`

Konfigurasi bayangan untuk *`ThingName`* untuk konfigurasi bayangan ini.

`classic`

(Opsional) Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik untuk `thingName` perangkat, setel ini ke`false`.

`namedShadows`

Daftar bayangan bernama yang ingin Anda sinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

`shadowDocuments`

(Opsional) Daftar bayangan perangkat tambahan untuk disinkronkan. Kami menyarankan Anda menggunakan `shadowDocumentsMap` parameter sebagai gantinya.

> **Note**
>
> Jika Anda menentukan `shadowDocuments` objek, Anda tidak harus menentukan `shadowDocumentsMap` objek.

Setiap objek dalam daftar ini berisi informasi berikut.

`thingName`

Nama objek perangkat untuk menyinkronkan bayangan.

`classic`

(Opsional) Jika Anda tidak ingin menyinkronkan bayangan perangkat klasik untuk `thingName` perangkat, setel ini ke`false`.

Default: `true`

`namedShadows`

(Opsional) Daftar bayangan perangkat bernama yang ingin Anda sinkronkan. Anda harus menentukan nama yang tepat dari bayangan.

`rateLimits`

(Opsional) Pengaturan yang menentukan batas tarif untuk permintaan layanan bayangan.

Objek ini berisi informasi berikut.

`maxOutboundSyncUpdatesPerSecond`

(Opsional) Jumlah maksimum permintaan sinkronisasi per detik yang ditransmisikan perangkat.

Default: 100 permintaan/detik

`maxTotalLocalRequestsRate`

(Opsional) Jumlah maksimum permintaan IPC lokal per detik yang dikirim ke perangkat inti.

Default: 200 permintaan/detik

`maxLocalRequestsPerSecondPerThing`

(Opsional) Jumlah maksimum permintaan IPC lokal per detik yang dikirim untuk setiap hal IoT yang terhubung.

Default: 20 permintaan/detik untuk setiap objek

> **Note**
>
> Parameter batas tingkat ini menentukan jumlah maksimum permintaan per detik untuk layanan bayangan lokal. Jumlah maksimum permintaan per detik untuk layanan AWS IoT Device Shadow tergantung pada Anda Wilayah AWS. Untuk informasi selengkapnya, lihat batasan untuk AWS IoT Device Shadow Service API di Referensi Umum Amazon Web.

`shadowDocumentSizeLimitBytes`

(Opsional) Ukuran maksimum yang diizinkan dari setiap dokumen status JSON untuk bayangan lokal.

Jika Anda meningkatkan nilai ini, Anda juga harus meningkatkan batas sumber daya untuk dokumen keadaan JSON untuk bayangan cloud. Untuk informasi selengkapnya, lihat batasan untuk AWS IoT Device Shadow Service API di Referensi Umum Amazon Web.

Default: 8192 byte

Maksimum: 30720 byte

Example Contoh: Pembaruan gabungan konfigurasi

Contoh berikut menunjukkan pembaruan gabungan konfigurasi sampel dengan semua parameter konfigurasi yang tersedia untuk komponen manajer bayangan.

```
{
  "synchronize": {
    "coreThing": {
      "classic": true,
```

```
      "namedShadows": [
        "MyCoreShadowA",
        "MyCoreShadowB"
      ]
    },
    "shadowDocuments": [
      {
        "thingName": "MyDevice1",
        "classic": false,
        "namedShadows": [
          "MyShadowA",
          "MyShadowB"
        ]
      },
      {
        "thingName": "MyDevice2",
        "classic": true,
        "namedShadows": []
      }
    ]
  },
  "rateLimits": {
    "maxOutboundSyncUpdatesPerSecond": 100,
    "maxTotalLocalRequestsRate": 200,
    "maxLocalRequestsPerSecondPerThing": 20
  },
  "shadowDocumentSizeLimitBytes": 8192
}
```

Berkas log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.3.8	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana shadow manager menciptakan situasi kebuntuan selama koneksi klien MQTT.
2.3.7	Perbaikan bug dan peningkatan • Memperbaiki masalah saat pengelola bayangan mencatat `NullPointerException` kesalahan secara berkala selama sinkronisasi pengelola bayangan.
2.3.6	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana properti bayangan yang dihapus melalui AWS Cloud pembaruan saat perangkat offline terus ada di bayangan lokal setelah mendapatkan kembali konektivitas.
2.3.5	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.

Versi	Perubahan
2.3.4	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk dokumen status bayangan nol dan kosong.
2.3.3	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.3.2	Perbaikan bug dan peningkatan • Memperbaiki masalah saat pengelola bayangan memasuki `BROKEN` status saat database bayangan lokal rusak. • Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.3.1	Perbaikan bug dan peningkatan • Memperbaiki kondisi yang dapat mencegah pembaruan cloud shadow dari sinkronisasi. • Memperbaiki masalah di mana perubahan konfigurasi sinkronisasi bayangan bernama hanya berlaku untuk satu bayangan bernama.
2.3.0	Perbaikan bug dan peningkatan • Memperbaiki masalah yang mungkin mencegah sinkronisasi bayangan saat kunci pribadi perangkat Greengrass disimpan dalam modul keamanan perangkat keras.
2.2.4	Perbaikan bug dan peningkatan • Memperbaiki masalah ketika validasi ukuran bayangan tidak konsisten dengan cloud saat memperbarui dokumen bayangan lokal. • Memperbaiki masalah saat pengelola bayangan berhenti mendengarkan pembaruan konfigurasi jika penerapan melakukan a `RESET` pada node konfigurasi.
2.2.3	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.2.2	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.2.1	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.

Versi	Perubahan
2.2.0	Fitur baru • Menambahkan dukungan untuk layanan bayangan lokal melalui antarmuka penerbitan/berlangganan lokal. Anda sekarang dapat berkomunikasi dengan broker pesan penerbitan/berlangganan lokal pada topik bayangan MQTT untuk mendapatkan, memperbarui, dan menghapus bayangan pada perangkat inti. Fitur ini memungkinkan Anda untuk menghubungkan perangkat klien ke layanan bayangan lokal dengan menggunakan jembatan MQTT untuk menyampaikan pesan tentang topik bayangan antara perangkat klien dan antarmuka penerbita n/berlangganan lokal. Fitur ini membutuhkan v2.6.0 atau yang lebih baru dari komponen inti Greengrass. Untuk menghubungkan perangkat klien ke layanan bayangan lokal, Anda juga harus menggunakan v2.2.0 atau yang lebih baru dari komponen jembatan MQTT. • Menambahkan `direction` opsi yang dapat Anda konfigurasikan untuk menyesuaikan arah untuk menyinkronkan bayangan antara layanan bayangan lokal dan AWS Cloud. Anda dapat mengonfigurasi opsi ini untuk mengurangi bandwidth dan koneksi ke file AWS Cloud.
2.1.1	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana kedalaman maksimum dalam `desired` dan `reported` bagian dokumen status bayangan perangkat JSON adalah 4 level, bukan 5 level. • Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.0	Fitur baru • Menambahkan dukungan untuk interval sinkronisasi bayangan periodik, sehingga Anda dapat mengonfigurasi perangkat inti untuk mengurangi penggunaan dan pengisian bandwidth.
2.0.6	Versi ini berisi perbaikan bug dan perbaikan.
2.0.5	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.

Versi	Perubahan
2.0.4	Perbaikan bug dan peningkatan • Memperbaiki masalah yang menyebabkan shadow manager menghapus versi yang baru dibuat dari bayangan apa pun yang sebelumnya dihapus. • Memperbarui operasi `DeleteThingShadow` IPC untuk menambah versi bayangan saat dipanggil.
2.0.3	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.2	Perbaikan bug dan peningkatan • Memperbaiki masalah yang menyebabkan manajer bayangan tidak mengenali properti `delta` ketika menyinkronkan bayangan keadaan dari AWS IoT Core. • Memperbaiki masalah yang terkadang menyebabkan permintaan sinkronisasi agar bayangan digabung secara tidak tepat.
2.0.1	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.0	Versi awal.

Amazon SNS

Komponen Amazon SNS (`aws.greengrass.SNS`) menerbitkan pesan ke topik Amazon Simple Notification Service (Amazon SNS). Anda dapat menggunakan komponen ini untuk mengirim peristiwa dari perangkat inti Greengrass ke server web, alamat email, dan pelanggan pesan lainnya. Untuk informasi lebih lanjut, lihat Apa itu Amazon SNS di Panduan Developer Amazon Simple Notification Service.

Untuk mempublikasikan topik Amazon SNS dengan komponen ini, mempublikasikan pesan ke topik di mana komponen ini berlangganan. Secara default, komponen ini berlangganan topik publikasi/berlangganan lokal `sns/message`. Anda dapat menentukan topik lain, termasuk topik AWS IoT Core MQTT, saat Anda menerapkan komponen ini.

Dalam komponen kustom Anda, Anda mungkin ingin menerapkan pemfilteran atau pemformatan logika untuk memproses pesan dari sumber lain sebelum Anda mempublikasikannya ke komponen

ini. Hal ini memungkinkan Anda untuk memusatkan logika pemrosesan pesan Anda pada satu komponen.

> **Note**
>
> Komponen ini menyediakan fungsionalitas yang mirip dengan konektor Amazon SNS di AWS IoT Greengrass V1. Untuk informasi selengkapnya, lihat konektor Amazon SNS dalam AWS IoT Greengrass Panduan Developer V1.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen Lambda () `aws.greengrass.lambda`. Inti Greengrass menjalankan fungsi Lambda komponen ini menggunakan komponen peluncur Lambda.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Anda harus memenuhi persyaratan untuk menjalankan fungsi Lambda. Jika Anda ingin perangkat inti untuk menjalankan fungsi Lambda kontainer, perangkat harus memenuhi persyaratan untuk melakukannya. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.
- Python versi 3.7 diinstal pada perangkat inti dan ditambahkan ke variabel lingkungan PATH.
- Topik Amazon SNS. Untuk informasi lebih lanjut, lihat Membuat topik Amazon SNS dalam Panduan Developer Amazon Simple Notification Service.
- Peran perangkat Greengrass harus mengizinkan tindakan `sns:Publish`, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

  ```
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": [
          "sns:Publish"
        ],
        "Effect": "Allow",
        "Resource": [
          "arn:aws:sns:region:account-id:topic-name"
        ]
      }
    ]
  }
  ```

  Anda dapat secara dinamis mengganti topik default dalam muatan pesan masukan untuk komponen ini. Jika aplikasi Anda menggunakan fitur ini, kebijakan IAM harus mencakup semua topik target sebagai sumber daya. Anda dapat memberikan akses terperinci atau bersyarat ke sumber daya (misalnya, dengan menggunakan skema penamaan wildcard *).
- Untuk menerima data keluaran dari komponen ini, Anda harus menggabungkan pemutakhiran konfigurasi berikut untuk komponen router langganan lama

(aws.greengrass.LegacySubscriptionRouter) saat menerapkan komponen ini. Konfigurasi ini menentukan topik di mana komponen ini menerbitkan tanggapan.

Legacy subscription router v2.1.x

```
{
  "subscriptions": {
    "aws-greengrass-sns": {
      "id": "aws-greengrass-sns",
      "source": "component:aws.greengrass.SNS",
      "subject": "sns/message/status",
      "target": "cloud"
    }
  }
}
```

Legacy subscription router v2.0.x

```
{
  "subscriptions": {
    "aws-greengrass-sns": {
      "id": "aws-greengrass-sns",
      "source": "arn:aws:lambda:region:aws:function:aws-greengrass-sns:version",
      "subject": "sns/message/status",
      "target": "cloud"
    }
  }
}
```

- Ganti *wilayah* dengan Wilayah AWS yang Anda gunakan.
- Ganti *versi* dengan versi fungsi Lambda yang komponen ini jalankan. Untuk menemukan versi fungsi Lambda, Anda harus melihat resep untuk versi komponen ini yang ingin Anda deploy. Buka halaman detail komponen ini di konsol AWS IoT Greengrass tersebut, dan cari pasangan nilai kunci fungsi Lambda. Pasangan kunci-nilai ini berisi nama dan versi fungsi Lambda.

> **Important**
>
> Anda harus memperbarui versi fungsi Lambda pada router langganan warisan setiap kali Anda men-deploy komponen ini. Hal ini memastikan bahwa Anda menggunakan versi fungsi Lambda yang benar untuk versi komponen yang Anda deploy.

Untuk informasi selengkapnya, lihat Buat deployment.

- Komponen Amazon SNS didukung untuk berjalan di VPC. Untuk menerapkan komponen ini di VPC, berikut ini diperlukan.
  - Komponen Amazon SNS harus memiliki konektivitas `sns.region.amazonaws.com` yang memiliki titik akhir VPC. `com.amazonaws.us-east-1.sns`

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Diperlukan	Deskripsi
`sns.region.amazonaws.com`	443	Ya	Mempublikasikan pesan ke Amazon SNS.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.7

Tabel berikut mencantumkan dependensi untuk versi 2.1.7 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.6

Tabel berikut mencantumkan dependensi untuk versi 2.1.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.5

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	^2.0.0	Keras

2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.3

Tabel berikut mencantumkan dependensi untuk versi 2.1.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.2

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.8 - 2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.0.8 dan 2.1.0 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.7

Tabel berikut mencantumkan dependensi untuk versi 2.0.7 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.6

Tabel berikut mencantumkan dependensi untuk versi 2.0.6 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.5

Tabel berikut mencantumkan dependensi untuk versi 2.0.5 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	^2.0.0	Keras

2.0.4

Tabel berikut mencantumkan dependensi untuk versi 2.0.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Keras
Peluncur Lambda	^2.0.0	Keras
Runtime Lambda	^2.0.0	Lunak
Layanan pertukaran Token	^2.0.0	Keras

2.0.3

Tabel berikut mencantumkan dependensi untuk versi 2.0.3 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Keras
Peluncur Lambda	>=1.0.0	Keras
Runtime Lambda	>=1.0.0	Lunak
Layanan pertukaran Token	>=1.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

> **Note**
>
> Konfigurasi default komponen ini meliputi parameter fungsi Lambda. Kami sarankan Anda mengedit hanya parameter berikut untuk mengonfigurasi komponen ini pada perangkat Anda.

`lambdaParams`

Sebuah objek yang berisi parameter untuk fungsi Lambda komponen ini. Objek ini berisi informasi berikut:

`EnvironmentVariables`

Sebuah objek yang berisi parameter fungsi Lambda ini. Objek ini berisi informasi berikut:

`DEFAULT_SNS_ARN`

ARN topik Amazon SNS default di mana komponen ini menerbitkan pesan. Anda dapat menimpa topik tujuan dengan properti `sns_topic_arn` dalam muatan pesan masukan.

`containerMode`

(Opsional) Mode kontainerisasi untuk komponen ini. Pilih dari salah satu pilihan berikut:

- `NoContainer` – Komponen tersebut tidak berjalan di lingkungan waktu aktif terisolasi.
- `GreengrassContainer`— Komponen berjalan di lingkungan runtime yang terisolasi di dalam AWS IoT Greengrass wadah.

Default: `GreengrassContainer`

`containerParams`

(Opsional) Sebuah objek yang berisi parameter kontainer untuk komponen ini. Komponen menggunakan parameter ini jika Anda menentukan `GreengrassContainer` untuk `containerMode`.

Objek ini berisi informasi berikut:

`memorySize`

(Opsional) Jumlah memori (dalam kilobyte) yang akan dialokasikan ke komponen.

Defaultnya 512 MB (525.312 KB).

`pubsubTopics`

(Opsional) Sebuah objek yang berisi topik di mana komponen berlangganan untuk menerima pesan. Anda dapat menentukan setiap topik dan apakah komponen berlangganan topik MQTT dari AWS IoT Core atau topik penerbitan/langganan lokal.

Objek ini berisi informasi berikut:

`0` - Ini adalah indeks himpunan sebagai string.

Objek yang berisi informasi berikut:

`type`

(Opsional) Jenis olahpesan publikasikan/berlangganan yang digunakan oleh komponen ini untuk berlangganan pesan. Pilih dari salah satu pilihan berikut:

- `PUB_SUB` — Berlangganan pesan publish/subscribe lokal. Jika Anda memilih opsi ini, topik tidak dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Pesan lokal publikasi/berlangganan.
- `IOT_CORE`— Berlangganan pesan AWS IoT Core MQTT. Jika Anda memilih opsi ini, topik dapat berisi wildcard MQTT. Untuk informasi lebih lanjut tentang cara mengirim pesan dari komponen kustom ketika Anda menentukan opsi ini, lihat Terbitkan/berlangganan pesan MQTT AWS IoT Core.

Default: `PUB_SUB`

`topic`

(Opsional) Topik yang menjadi langganan komponen untuk menerima pesan. Jika Anda menentukan `IotCore` untuk `type`, Anda dapat menggunakan wildcard MQTT (+ dan #) dalam topik ini.

Example Contoh: Pembaruan gabungan konfigurasi (mode kontainer)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_SNS_ARN": "arn:aws:sns:us-west-2:123456789012:mytopic"
    }
  },
  "containerMode": "GreengrassContainer"
```

```
}
```

Example Contoh: Pembaruan gabungan konfigurasi (tidak ada mode kontainer)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_SNS_ARN": "arn:aws:sns:us-west-2:123456789012:mytopic"
    }
  },
  "containerMode": "NoContainer"
}
```

Data input

Komponen ini menerima pesan pada topik berikut dan menerbitkan pesan sebagaimana adanya ke target topik Amazon SNS. Secara default, komponen ini berlangganan topik publikasi/berlangganan lokal. Untuk informasi lebih lanjut tentang cara mempublikasikan pesan ke komponen ini dari komponen kustom Anda, lihat Pesan lokal publikasi/berlangganan.

Topik default (publish/subscribe lokal): `sns/message`

Pesan menerima properti berikut. Pesan input harus dalam format JSON.

`request`

Informasi tentang pesan yang akan dikirim ke topik Amazon SNS.

Jenis: `object` yang berisi informasi berikut:

`message`

Isi dari pesan sebagai string.

Untuk mengirim objek JSON, buat serialnya sebagai string, dan tentukan `json` untuk properti `message_structure`.

Tipe: `string`

`subject`

(Opsional) Subjek pesan.

Tipe: `string`

Subjek dapat berupa teks ASCII dan hingga 100 karakter. Ia harus dimulai dengan huruf, angka, atau tanda baca. Ia tidak dapat berupa jeda baris atau karakter kontrol.

`sns_topic_arn`

ARN topik Amazon SNS tempat komponen ini menerbitkan pesan. Tentukan properti ini untuk menimpa topik Amazon SNS default.

Tipe: `string`

`message_structure`

(Opsional) Struktur pesan. Tentukan `json` untuk mengirim pesan JSON yang Anda buat serialnya sebagai string dalam properti `content`.

Tipe: `string`

Nilai yang valid: `json`

`id`

ID acak untuk permintaan. Gunakan properti ini untuk memetakan permintaan inpun untuk respons output. Ketika Anda menentukan properti ini, komponen menetapkan properti `id` di objek respons untuk nilai ini.

Tipe: `string`

> **Note**
>
> Ukuran pesan dapat maksimal 256 KB.

Example Contoh input: Pesan string

```
{
  "request": {
    "subject": "Message subject",
    "message": "Message data",
    "sns_topic_arn": "arn:aws:sns:region:account-id:topic2-name"
  },
  "id": "request123"
}
```

Example Contoh input: pesan JSON

```
{
  "request": {
    "subject": "Message subject",
    "message": "{ \"default\": \"Message data\" }",
    "message_structure": "json"
  },
  "id": "request123"
}
```

Data output

Komponen ini menerbitkan tanggapan sebagai data output pada topik MQTT berikut secara default. Anda harus menentukan topik ini sebagai `subject` dalam konfigurasi untuk komponen router langganan warisan. Untuk informasi lebih lanjut tentang cara mempublikasikan pesan ke komponen ini dari komponen kustom Anda, lihat Terbitkan/berlangganan pesan MQTT AWS IoT Core.

Topik default (AWS IoT Core MQTT): `sns/message/status`

Example Contoh output: Berhasil

```
{
  "response": {
    "sns_message_id": "f80a81bc-f44c-56f2-a0f0-d5af6a727c8a",
    "status": "success"
  },
  "id": "request123"
}
```

Example Contoh output: Gagal

```
{
  "response" : {
    "error": "InvalidInputException",
    "error_message": "SNS Topic Arn is invalid",
    "status": "fail"
  },
  "id": "request123"
}
```

Berkas log lokal

Komponen ini menggunakan file log berikut.

```
/greengrass/v2/logs/aws.greengrass.SNS.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* dengan jalur ke folder AWS IoT Greengrass root.

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

Lisensi

Komponen ini 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/MIT License

Komponen ini dirilis menurut Perjanjian Lisensi Perangkat Lunak Greengrass Core.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.7	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.6	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.

Versi	Perubahan
2.1.5	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.4	Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.3	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.
2.1.2	Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.1.1	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.1.0	Fitur baru • Menambahkan dukungan untuk konfigurasi proxy jaringan HTTPS. Lihat informasi yang lebih lengkap di Hubungkan pada port 443 atau melalui proksi jaringan dan Aktifkan perangkat inti untuk mempercayai proxy HTTPS.
2.0.8	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.7	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.6	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.5	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.0.4	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.
2.0.3	Versi awal.

Manajer pengaliran

Komponen pengelola aliran (`aws.greengrass.StreamManager`) memungkinkan Anda memproses aliran data untuk ditransfer ke perangkat inti AWS Cloud Greengrass.

Untuk informasi lebih lanjut tentang cara mengonfigurasi dan menggunakan stream manager di komponen kustom, lihat Kelola aliran data di perangkat inti Greengrass.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.1.x
- 2.0.x

> **Note**
>
> Jika Anda menggunakan pengelola aliran untuk mengekspor data ke cloud, Anda tidak dapat memutakhirkan versi 2.0.7 komponen pengelola aliran ke versi antara v2.0.8 dan v2.0.11. Jika Anda menerapkan pengelola aliran untuk pertama kalinya, kami sangat menyarankan agar Anda menerapkan versi terbaru komponen pengelola aliran.

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Peran pertukaran token harus mengizinkan akses ke AWS Cloud tujuan yang Anda gunakan dengan pengelola aliran. Lihat informasi yang lebih lengkap di:
  - the section called “Saluran AWS IoT Analytics”
  - the section called “Amazon Kinesis data streams”
  - the section called “Properti aset AWS IoT SiteWise”
  - the section called “Objek Amazon S3”
- Komponen pengelola aliran didukung untuk berjalan di VPC. Untuk menerapkan komponen ini di VPC, berikut ini diperlukan.
  - Komponen pengelola aliran harus memiliki konektivitas ke AWS layanan tempat Anda mempublikasikan data.
    - Amazon S3: `com.amazonaws.region.s3`
    - Aliran Data Amazon Kinesis: `com.amazonaws.region.kinesis-streams`
    - AWS IoT SiteWise: `com.amazonaws.region.iotsitewise.data`
  - Jika Anda mempublikasikan data ke Amazon S3 di `us-east-1` wilayah tersebut, komponen ini akan mencoba menggunakan titik akhir global S3 secara default; namun, titik akhir ini tidak tersedia melalui titik akhir antarmuka VPC Amazon S3. Untuk informasi selengkapnya, lihat Pembatasan dan batasan AWS PrivateLink untuk Amazon S3. Untuk mengatasi ini, Anda dapat memilih dari opsi berikut.
    - Konfigurasikan komponen pengelola aliran untuk menggunakan titik akhir S3 regional di `us-east-1` wilayah tersebut, dengan menyediakan variabel `AWS_S3_US_EAST_1_REGIONAL_ENDPOINT=regional` lingkungan.
    - Buat titik akhir VPC gateway Amazon S3 alih-alih titik akhir VPC antarmuka Amazon S3. Titik akhir gateway S3 mendukung akses ke titik akhir global S3. Untuk informasi selengkapnya, lihat Membuat titik akhir gateway.

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Diperlukan	Deskripsi
`iotanalytics. region.amazonaws.com`	443	Tidak	Diperlukan jika Anda mempublikasikan data ke AWS IoT Analytics.
`kinesis.region.amazonaws.com`	443	Tidak	Diperlukan jika Anda mempublikasikan data ke Firehose.
`data.iotsitewise. region.amazonaws.com`	443	Tidak	Diperlukan jika Anda mempublikasikan data ke AWS IoT SiteWise.
`*.s3.amazonaws.com`	443	Tidak	Diperlukan jika Anda mempublikasikan data ke bucket S3. Anda dapat mengganti * dengan nama setiap

Titik Akhir	Port	Diperlukan	Deskripsi
			bucket tempat Anda mempublikasikan data.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

2.1.11

Tabel berikut mencantumkan dependensi untuk versi 2.1.11 hingga 2.1.10 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.13.0	Lunak
Layanan penukaran Token	>=0.0.0	Keras

2.1.9 – 2.1.10

Tabel berikut mencantumkan dependensi untuk versi 2.1.9 hingga 2.1.10 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.12.0	Lunak
Layanan penukaran Token	>=0.0.0	Keras

2.1.5 – 2.1.8

Tabel berikut mencantumkan dependensi untuk versi 2.1.5 hingga 2.1.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.11.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.1.2 – 2.1.4

Tabel berikut mencantumkan dependensi untuk versi 2.1.2 hingga 2.1.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.10.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.1.1

Tabel berikut mencantumkan dependensi untuk versi 2.1.1 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.9.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.1.0

Tabel berikut mencantumkan dependensi untuk versi 2.1.0 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.8.0	Lunak

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	>=0.0.0	Keras

2.0.15

Tabel berikut mencantumkan dependensi untuk versi 2.0.15 dari komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.7.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.0.13 and 2.0.14

Tabel berikut mencantumkan dependensi untuk versi 2.0.13 dan 2.0.14 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.6.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.0.11 and 2.0.12

Tabel berikut mencantumkan dependensi untuk versi 2.0.11 dan 2.0.12 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.5.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.0.10

Tabel berikut mencantumkan dependensi untuk versi 2.0.10 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.4.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.0.9

Tabel berikut mencantumkan dependensi untuk versi 2.0.9 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.3.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.0.8

Tabel berikut mencantumkan dependensi untuk versi 2.0.8 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.0 <2.2.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

2.0.7

Tabel berikut mencantumkan dependensi untuk versi 2.0.7 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.0.3 <2.1.0	Lunak
Layanan pertukaran Token	>=0.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`STREAM_MANAGER_STORE_ROOT_DIR`

(Opsional) Path absolut dari direktori lokal yang digunakan untuk menyimpan pengaliran. Nilai ini harus dimulai dengan garis miring ke depan (misalnya, `/data`).

Anda harus menentukan folder yang ada, dan pengguna sistem yang menjalankan komponen manajer aliran harus memiliki izin untuk membaca dan menulis ke folder ini. Misalnya, Anda dapat menjalankan perintah berikut untuk membuat dan mengkonfigurasi folder`/var/greengrass/streams`, yang Anda tentukan sebagai folder root stream manager. Perintah ini memungkinkan pengguna sistem default,`ggc_user`, untuk membaca dan menulis ke folder ini.

```
sudo mkdir /var/greengrass/streams
sudo chown ggc_user /var/greengrass/streams
sudo chmod 700 /var/greengrass/streams
```

Default: *`/greengrass/v2`*`/work/aws.greengrass.StreamManager`

`STREAM_MANAGER_SERVER_PORT`

(Opsional) Nomor port lokal yang akan digunakan untuk berkomunikasi dengan stream manager.

Anda dapat menentukan `0` untuk menggunakan port yang tersedia secara acak.

Default: `8088`

`STREAM_MANAGER_AUTHENTICATE_CLIENT`

(Opsional) Anda dapat membuatnya wajib bagi klien untuk melakukan autentikasi sebelum mereka dapat berinteraksi dengan stream manager. Stream Manager SDK mengontrol interaksi antara klien dan manajer aliran. Parameter ini menentukan klien mana yang dapat memanggil Manajer Pengaliran untuk bekerja dengan pengaliran. Untuk informasi selengkapnya, lihat autentikasi klien stream manager.

Jika Anda menentukan`true`, Stream Manager SDK hanya mengizinkan komponen Greengrass sebagai klien.

Jika Anda menentukan`false`, Stream Manager SDK memungkinkan semua proses pada perangkat inti menjadi klien.

Default: `true`

`STREAM_MANAGER_EXPORTER_MAX_BANDWIDTH`

(Opsional) Bandwidth maksimum rata-rata (dalam kilobit per detik) yang dapat digunakan oleh pengelola pengaliran untuk mengekspor data.

Default: Tanpa batas

`STREAM_MANAGER_EXPORTER_THREAD_POOL_SIZE`

(Opsional) Jumlah maksimum utas aktif yang dapat digunakan stream manager untuk mengekspor data.

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 aliran mengonsumsi sumber daya perangkat keras, jadi cobalah untuk membatasi jumlah aliran ekspor pada perangkat yang dibatasi.

Default: 5 utas

`STREAM_MANAGER_EXPORTER_S3_DESTINATION_MULTIPART_UPLOAD_MIN_PART_SIZE_BYTES`

(Opsional) 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.

> **Note**
>
> Stream manager menggunakan properti aliran `sizeThresholdForMultipartUploadBytes` untuk menentukan apakah akan mengekspor ke Amazon S3 sebagai unggahan tunggal atau multipart. Komponen AWS IoT Greengrass dapat mengatur ambang batas ini ketika membuat aliran yang diekspor ke Amazon S3.

Default: `5242880` (5 MB). Ini juga merupakan nilai minimum.

`LOG_LEVEL`

(Opsional) Tingkat logging untuk komponen. Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `TRACE`
- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

`JVM_ARGS`

(Opsional) Argumen Mesin Virtual Java kustom yang akan disampaikan ke stream manager saat startup. Pisahkan beberapa argumen dengan 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.

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk menggunakan port non-default.

```
{
  "STREAM_MANAGER_SERVER_PORT": "18088"
}
```

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.greengrass.StreamManager.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.StreamManager.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

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

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.StreamManager.log -Tail 10 -
  Wait
  ```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.1.12	Perbaikan bug dan peningkatan Memperbarui urutan kredensi yang digunakan sehingga kredenal Greengrass lebih disukai untuk permintaan layanan. AWS
2.1.11	Versi diperbarui untuk Greengrass nucleus versi 2.12.0 rilis.
2.1.10	Perbaikan bug dan peningkatan Memperbaiki masalah di mana konfigurasi proxy HTTPS tidak mempercay ai rantai sertifikat otoritas sertifikat Greengrass Certificate Authority (CA).
2.1.9	Versi diperbarui untuk Greengrass nucleus versi 2.11.0 rilis.
2.1.8	Perbaikan bug dan peningkatan Memperbaiki masalah di mana manajer aliran mencoba ulang SiteWise ekspor yang gagal tanpa batas. `InvalidRequestException`

Versi	Perubahan
2.1.7	Perbaikan bug dan peningkatan Memperbaiki masalah di mana manajer aliran gagal membaca konfigurasi proxy dengan benar.
2.1.6	Perbaikan bug dan peningkatan Memperbaiki masalah yang dapat menyebabkan crash saat startup pada prosesor ARMv8 tertentu, termasuk Jetson Nano.
2.1.5	Versi diperbarui untuk Greengrass nucleus versi 2.10.0 rilis.
2.1.4	Perbaikan bug dan peningkatan • Memperbaiki masalah saat entri untuk aset properti yang sama dengan stempel waktu yang sama dalam satu batch kembali `ConflictingOperationException` dari SiteWise API yang menyebabkan pengelola aliran terus mencoba lagi. • Memperbarui batas waktu koneksi default dari 3 detik hingga 1 menit.
2.1.3	Perbaikan bug dan peningkatan Memperbaiki masalah startup pada OS Windows saat berjalan sebagai pengguna SYSTEM.
2.1.2	Perbaikan bug dan peningkatan • Memperbaiki masalah pada OS Windows yang menggunakan bahasa non-Inggris. • Versi diperbarui untuk Greengrass nucleus versi 2.9.0 rilis.
2.1.1	Versi diperbarui untuk Greengrass nucleus versi 2.8.0 rilis.

Versi	Perubahan
2.1.0	Fitur baru • Memperbarui komponen ini untuk secara otomatis mengirim metrik telemetri ke Amazon. EventBridge Untuk informasi selengkapnya, lihat Kumpulkan data telemetri kondisi sistem dari perangkat inti AWS IoT Greengrass. Fitur ini membutuhkan v2.7.0 atau yang lebih baru dari komponen inti Greengrass. • Versi diperbarui untuk Greengrass nucleus versi 2.7.0 rilis.
2.0.15	Versi diperbarui untuk Greengrass nucleus versi 2.6.0 rilis.
2.0.14	Versi ini berisi perbaikan bug dan perbaikan.
2.0.13	Versi diperbarui untuk Greengrass nucleus versi 2.5.0 rilis.
2.0.12	Perbaikan bug dan peningkatan Memperbaiki masalah yang mencegah upgrade stream manager v2.0.7 ke versi antara v2.0.8 dan v2.0.11. Jika Anda menggunakan pengelola aliran untuk mengekspor data ke cloud, Anda sekarang dapat meningkatkan ke v2.0.12.
2.0.11	Versi diperbarui untuk Greengrass nucleus versi 2.4.0 rilis.
2.0.10	Versi yang diperbarui untuk rilis inti Greengrass versi 2.3.0.
2.0.9	Versi yang diperbarui untuk rilis inti Greengrass versi 2.2.0.
2.0.8	Versi yang diperbarui untuk rilis inti Greengrass versi 2.1.0.
2.0.7	Versi awal.

Agen Systems Manager

Komponen AWS Systems Manager Agen (`aws.greengrass.SystemsManagerAgent`) menginstal Agen Systems Manager, sehingga Anda dapat mengelola perangkat inti dengan Systems Manager.

Systems Manager adalah AWS layanan yang dapat Anda gunakan untuk melihat dan mengontrol infrastruktur AWS, termasuk instans Amazon EC2, server lokal dan mesin virtual (VM), dan perangkat edge. Systems Manager memungkinkan Anda untuk melihat data operasional, mengotomatiskan tugas operasi, dan menjaga keamanan dan kepatuhan. Untuk informasi lebih lanjut, lihat Apa itu AWS Systems Manager? dan Tentang Agen Systems Manager di Panduan AWS Systems Manager Pengguna.

Alat dan fitur Systems Manager disebut kemampuan. Perangkat inti Greengrass mendukung semua kemampuan Systems Manager. Untuk informasi selengkapnya tentang kemampuan ini dan cara menggunakan Systems Manager untuk mengelola perangkat inti, lihat kemampuan Systems Manager di Panduan AWS Systems Manager Pengguna.

Topik

Versi

Komponen ini memiliki versi berikut:

- 1.1.x
- 1.0.x

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini hanya dapat diinstal pada perangkat inti Linux.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Greengrass yang berjalan pada platform Linux 64-bit: Armv8 (AArch64) atau x86_64.
- Anda harus memiliki peran layanan AWS Identity and Access Management (IAM) yang dapat diasumsikan oleh Systems Manager. Peran ini harus menyertakan kebijakan ManagedInstanceCore terkelola AmazonSSM atau kebijakan khusus yang menentukan izin yang setara. Untuk informasi selengkapnya, lihat Membuat peran layanan IAM untuk perangkat edge di Panduan AWS Systems Manager Pengguna.

  Saat menerapkan komponen ini, Anda harus menentukan nama peran ini untuk parameter `SSMRegistrationRole` konfigurasi.
- Peran perangkat Greengrass harus memungkinkan dan tindakan. `ssm:AddTagsToResource` `ssm:RegisterManagedInstance` Peran perangkat juga harus memungkinkan `iam:PassRole` tindakan untuk peran layanan IAM yang memenuhi persyaratan sebelumnya. Contoh berikut kebijakan IAM memberikan izin ini.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::account-id:role/SSMServiceRole"
      ]
    },
    {
      "Action": [
        "ssm:AddTagsToResource",
        "ssm:RegisterManagedInstance"
      ],
```

```
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Wajib	Deskripsi
`ec2messages.` *region*`.amazonaws.com`	443	Ya	Berkomunikasi dengan layanan Systems Manager di AWS Cloud.
`ssm.`*region*`.amazonaws.com`	443	Ya	Daftarkan perangkat inti sebagai node terkelola Systems Manager.
`ssmmessages.` *region*`.amazonaws.com`	443	Ya	Berkomunikasi dengan Session Manager, kemampuan Systems

Titik Akhir	Port	Wajib	Deskripsi
			Manager, di AWS Cloud.

Untuk informasi selengkapnya, lihat Referensi: ec2messages, ssmmessages, dan panggilan API lainnya di Panduan Pengguna.AWS Systems Manager

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

Tabel berikut mencantumkan dependensi untuk versi 1.0.0 hingga 1.2.4 komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	^2.0.0	Lunak

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini menyediakan parameter konfigurasi berikut yang dapat Anda sesuaikan ketika Anda men-deploy komponen.

`SSMRegistrationRole`

Peran layanan IAM yang dapat diasumsikan oleh Systems Manager dan yang mencakup kebijakan ManagedInstanceCore terkelola AmazonSSM atau kebijakan kustom yang mendefinisikan izin yang setara. Untuk informasi selengkapnya, lihat Membuat peran layanan IAM untuk perangkat edge di Panduan AWS Systems Manager Pengguna.

`SSMOverrideExistingRegistration`

(Opsional) Jika perangkat inti sudah menjalankan Agen Systems Manager yang terdaftar dengan aktivasi hibrida, Anda dapat mengganti pendaftaran Agen Systems Manager perangkat yang ada. Setel opsi ini `true` untuk mendaftarkan perangkat inti sebagai node terkelola menggunakan Agen Systems Manager yang disediakan komponen ini.

> **Note**
>
> Opsi ini hanya berlaku untuk perangkat yang terdaftar dengan aktivasi hibrida. Jika perangkat inti berjalan pada instans Amazon EC2 dengan Agen Systems Manager diinstal dan peran profil instans dikonfigurasi, ID node terkelola instans Amazon EC2 yang ada akan dimulai dengan. `i-` Saat Anda menginstal komponen Systems Manager Agent, agen Systems Manager mendaftarkan node terkelola baru yang ID-nya dimulai dengan `mi-` alih-alih. `i-` Kemudian, Anda dapat menggunakan node terkelola yang IDnya dimulai `mi-` untuk mengelola perangkat inti dengan Systems Manager.

Default: `false`

`SSMResourceTags`

(Opsional) Tag untuk ditambahkan ke node terkelola Systems Manager yang dibuat komponen ini untuk perangkat inti. Anda dapat menggunakan tag ini untuk mengelola grup perangkat inti dengan Systems Manager. Misalnya, Anda dapat menjalankan perintah di semua perangkat yang memiliki tag yang Anda tentukan.

Tentukan daftar di mana setiap tag adalah objek dengan a `Key` dan a`Value`. Misalnya, nilai berikut untuk `SSMResourceTags` menginstruksikan komponen ini untuk menyetel **`Owner`** tag **`richard-roe`** pada node terkelola perangkat inti.

```
[
  {
    "Key": "Owner",
    "Value": "richard-roe"
  }
]
```

Komponen ini mengabaikan tag ini jika node terkelola sudah ada dan `SSMOverrideExistingRegistration` sudah ada. `false`

Example Contoh: Pembaruan gabungan konfigurasi

Contoh konfigurasi berikut menentukan untuk menggunakan peran layanan bernama `SSMServiceRole` untuk memungkinkan perangkat inti untuk mendaftar dan berkomunikasi dengan Systems Manager.

```
{
  "SSMRegistrationRole": "SSMServiceRole",
  "SSMOverrideExistingRegistration": false,
  "SSMResourceTags": [
    {
      "Key": "Owner",
      "Value": "richard-roe"
    },
    {
      "Key": "Team",
      "Value": "solar"
    }
  ]
}
```

File log lokal

Perangkat lunak Systems Manager Agent menulis log ke folder di luar folder root Greengrass. Untuk informasi selengkapnya, lihat Melihat log Agen Systems Manager di Panduan AWS Systems Manager Pengguna.

Komponen Systems Manager Agent menggunakan skrip shell untuk menginstal, memulai, dan menghentikan Agen Systems Manager. Anda dapat menemukan output dari skrip ini di file log berikut.

```
/greengrass/v2/logs/aws.greengrass.SystemsManagerAgent.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* dengan jalur ke folder AWS IoT Greengrass root.

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

Lihat juga

- Kelola perangkat inti Greengrass dengan AWS Systems Manager
- Apa itu AWS Systems Manager? dalam AWS Systems Manager Panduan Penggguna
- Tentang Agen Systems Manager di Panduan AWS Systems Manager Pengguna

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
1.2.4	Perbaikan bug dan perbaikan Memperbarui komponen ini untuk mendapatkan versi Agen 3.2.2303.0.
1.2.3	Perbaikan bug dan peningkatan • Menambahkan percobaan ulang untuk instalasi komponen Agen dengan snap di Greengrass. • Memperbarui konfigurasi komponen Agen untuk hanya menggunakan Identitas Onprem di Greengrass. • Memperbarui komponen ini untuk memperbarui Agen hanya jika versi Agen yang diinstal tidak cocok dengan versi komponen Greengrass SSM Agent.
1.1.0	Versi ini berisi perbaikan bug dan perbaikan.
1.0.0	Versi awal.

Layanan pertukaran token

Komponen layanan pertukaran token (`aws.greengrass.TokenExchangeService`) menyediakan kredensial yang dapat Anda gunakan untuk berinteraksi dengan layanan AWS dalam komponen AWS kustom Anda.

Layanan pertukaran token menjalankan instans kontainer Amazon Elastic Container Service (Amazon ECS) sebagai server lokal. Server lokal ini terhubung ke penyedia

kredensial AWS IoT menggunakan alias peran AWS IoT yang Anda konfigurasikan dalam komponen nukleus Greengrass inti. Komponen ini menyediakan dua variabel lingkungan, `AWS_CONTAINER_CREDENTIALS_FULL_URI` dan`AWS_CONTAINER_AUTHORIZATION_TOKEN`. `AWS_CONTAINER_CREDENTIALS_FULL_URI`mendefinisikan URI ke server lokal ini. Saat komponen membuat klien AWS SDK, klien mengenali variabel lingkungan URI ini dan menggunakan token di dalamnya `AWS_CONTAINER_AUTHORIZATION_TOKEN` untuk terhubung ke layanan pertukaran token dan mengambil AWS kredensialnya. Hal ini memungkinkan perangkat inti Greengrass untuk memanggil operasi layanan AWS. Untuk informasi lebih lanjut tentang cara menggunakan komponen dalam komponen kustom Anda, lihat Berinteraksilah dengan layanan AWS.

> **Important**
>
> Dukungan untuk memperoleh kredensial AWS dengan cara ini telah ditambahkan ke SDK AWS pada 13 Juli 2016. Komponen Anda harus menggunakan Versi SDK AWS yang dibuat pada atau setelah tanggal tersebut. Untuk informasi lebih lanjut, lihat Menggunakan SDK AWS yang didukung pada Panduan Developer Layanan Amazon Elastic Container.

Topik

Versi

Komponen ini memiliki versi berikut:

- 2.0.x

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Dependensi

Komponen ini tidak memiliki dependensi apa pun.

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

File log lokal

Komponen ini menggunakan file log yang sama dengan komponen inti Greengrass.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

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

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

Linux

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

Windows (PowerShell)

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

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.0.3	Versi awal.

Kolektor IoT SiteWise OPC-UA

Komponen kolektor IoT SiteWise OPC-UA (`aws.iot.SiteWiseEdgeCollectorOpcua`) memungkinkan AWS IoT SiteWise gateway untuk mengumpulkan data dari server OPC-UA lokal.

Dengan komponen ini, AWS IoT SiteWise gateway dapat terhubung ke beberapa server OPC-UA. Untuk informasi selengkapnya tentang AWS IoT SiteWise gateway, lihat Menggunakan AWS IoT SiteWise di tepi di AWS IoT SiteWise Panduan Pengguna.

Topik

- Versi
- Tipe
- Sistem operasi
- Persyaratan
- Dependensi
- Konfigurasi
- Data input
- Data output

- Berkas log lokal
- Pemecahan masalah dan debugging
- Lisensi
- Changelog
- Lihat juga

Versi

Komponen ini memiliki versi berikut:

- 2.4.x
- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Greengrass harus berjalan di salah satu platform berikut:

- OS: Ubuntu 18.04 atau yang lebih baru

  Arsitektur: x86_64 (AMD64) atau ARMv8 (Aarch64)
- OS: Perusahaan Red Hat Linux (RHEL) 8

  Arsitektur: x86_64 (AMD64) atau ARMv8 (Aarch64)
- OS: Amazon Linux 2

  Arsitektur: x86_64 (AMD64) atau ARMv8 (Aarch64)
- OS: Debian 11

  Arsitektur: x86_64 (AMD64) atau ARMv8 (Aarch64)
- OS: Windows Server 2019 atau yang lebih baru

  Arsitektur: x86_64 (AMD64)
- Perangkat inti Greengrass harus memungkinkan konektivitas jaringan keluar ke server OPC-UA.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

Tabel berikut mencantumkan dependensi untuk semua versi komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.3.0 <3.0.0	Keras
Manajer pengaliran	> 2.0.10 <3.0.0	Keras
Secrets manager	>=2.0.8 <3.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

Anda dapat menggunakan AWS IoT SiteWise konsol atau API untuk mengonfigurasi komponen kolektor IoT SiteWise OPC-UA. Untuk informasi selengkapnya, lihat Langkah 4: Menambahkan sumber data - opsional di Panduan AWS IoT SiteWise Pengguna.

Data input

Komponen ini hanya menerima data dalam format berikut, semua yang lain akan diabaikan dan dibuang. Tabel di bawah ini memetakan tipe data OPC UA ke SiteWise padanannya.

SiteWise tipe data	Tipe data OPC UA	Deskripsi

`STRING`	`String`
	`Guid`
	`XmlElement`
		Sebuah string panjang maksimum 1024 byte.

`INTEGER`	`SByte`
	`Byte`
	`Int16`
	`UInt16`
	`Int32`
	`UInt32*`
	`Int64*`
		Integer 32-bit yang ditandatangani dengan rentang dari. `-2,147,483,648 to 2,147,483,647`

`DOUBLE`	`UInt32*`
	`Int64*`
	`Float`
	`Double`
		Nomor floating point dengan rentang dari `-10^100 to 10^100` dan presisi `IEEE 754` ganda.

SiteWise tipe data	Tipe data OPC UA	Deskripsi
`BOOLEAN`	`Boolean`	`true` atau `false`.

* Untuk tipe data OPC UA `UInt32` dan`Int64`, tipe SiteWise datanya akan `INTEGER` jika SiteWise mampu mewakili nilainya, jika tidak maka akan menjadi`DOUBLE`.

Data output

Komponen ini menulis `BatchPutAssetPropertyValue` pesan ke manajer AWS IoT Greengrass streaming. Untuk informasi selengkapnya, lihat BatchPutAssetPropertyValue di dalam Referensi API AWS IoT SiteWise .

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
  ```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log -Tail
 10 -Wait
```

Pemecahan masalah dan debugging

Komponen ini mencakup log peristiwa baru untuk membantu pelanggan mengidentifikasi dan memperbaiki masalah. File log terpisah dari file log lokal, dan ditemukan di lokasi berikut. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua/logs/
IotSiteWiseOpcUaCollectorEvents.log
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgeCollectorOpcua\logs
\IotSiteWiseOpcUaCollectorEvents.log
```

Log ini mencakup informasi terperinci dan instruksi pemecahan masalah. Informasi pemecahan masalah disediakan bersama diagnostik, dengan deskripsi tentang cara memperbaiki masalah, dan terkadang dengan tautan ke informasi lebih lanjut. Informasi diagnostik meliputi:

- Tingkat kepelikan
- Stempel Waktu
- Informasi khusus acara tambahan

Example Contoh log

```
dataSourceConnectionSuccess:
  Summary: Successfully connected to OpcUa server
  Level: INFO
  Timestamp: '2023-06-15T21:04:16.303Z'
  Description: Successfully connected to the data source.
```

```
  AssociatedMetrics:
  - Name: FetchedDataStreams
    Description: The number of fetched data streams for this data source
    Value: 1.0
    Namespace: IoTSiteWise
    Dimensions:
    - Name: SourceName
      Value: SourceName{value=OPC-UA Server}
    - Name: ThingName
      Value: test-core
  AssociatedData:
  - Name: DataSourceTrace
    Description: Name of the data source
    Data:
    - OPC-UA Server
  - Name: EndpointUri
    Description: The endpoint to which the connection was attempted.
    Data:
    - '"opc.tcp://10.0.0.1:1234"'
```

Lisensi

Komponen ini dirilis menurut Perjanjian Lisensi Perangkat Lunak Greengrass Core.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
2.4.2	Perbaikan bug dan peningkatan • Memperbaiki masalah selama penemuan server OPC UA di mana node dapat ditemukan beberapa kali. • Memperbaiki fitur snapshot untuk memastikan stempel waktu baru untuk setiap titik data snapshot.
2.4.1	Perbaikan bug dan peningkatan • Memperbaiki masalah yang terkait dengan dukungan proxy. • Memperbaiki masalah saat pembersihan utas gagal dan menyebabkan penyumbatan data.

Versi	Perubahan
2.4.0	Fitur baru • Menambahkan log peristiwa untuk membuatnya lebih mudah untuk mengidentifikasi dan memperbaiki masalah. Perbaikan bug dan peningkatan • Memperbaiki masalah dengan klien OPC-UA yang menyebabkan kesalahan sertifikat saat menghubungkan ke server OPC-UA yang menggunakan spesifikasi OPC-UA versi 1.05.
2.3.0	Fitur baru • Menambahkan dukungan untuk konfigurasi proxy HTTP inti Greengrass di Linux. Perbaikan bug dan peningkatan • Memperbaiki masalah keamanan (CVE-2019-19135).
2.2.0	Fitur baru • Menambahkan dukungan untuk menginstal Data Collection Pack pada arsitektur Linux ARMv8. • Persyaratan minimum untuk Linux ARMv8: • Memori: 4 GB • CPU: ARM Cortex-A72 atau spesifikasi yang setara Perbaikan bug dan peningkatan • Meningkatkan pencatatan metrik dalam proses penemuan simpul. • Meningkatkan penanganan tipe data yang tidak didukung. • Meningkatkan pencatatan kesalahan aliran data.

Versi	Perubahan
2.1.3	Fitur baru • Menambahkan dukungan untuk Windows Server 2019 atau lebih tinggi. Perbaikan bug dan peningkatan • Memperbaiki pesan galat saat Anda menerapkan komponen ini pada perangkat yang tidak didukung.

2.1.1	Fitur baru
	• Menambahkan dukungan untuk mengonfigurasi properti berlangganan berikut:
	  • DataChangeTrigger- Anda dapat menentukan kondisi yang memulai peringatan perubahan data.
	  • QueueSize- Kedalaman antrian pada server OPC-UA untuk metrik tertentu di mana pemberitahuan untuk Item yang Dipantau diantrian.
	  • PublishingIntervalMilliseconds- Interval (dalam milidetik) dari siklus penerbitan yang ditentukan saat langganan dibuat.
	  • SnapshotFrequencyMilliseconds - Anda dapat mengonfigurasi pengaturan batas waktu frekuensi snapshot untuk memastikan bahwa AWS IoT SiteWise Edge menyerap aliran data yang stabil.
	• Versi ini mendukung konsumsi data BAD berkualitas dan menyaring data berdasarkan kualitas data berikut:
	  • `UNCERTAIN` data berkualitas
	  • `BAD`data berkualitas
	Perbaikan bug dan peningkatan
	• Perbaikan metrik pelanggan.
	• Memperbaiki pengkodean keamanan yang terkadang menyebabkan masalah saat menghubungkan ke server dengan enkripsi diaktifkan.
	• Memperbaiki masalah di mana grup properti gagal memperbarui.

2.0.3	Perbaikan bug dan perbaikan.
2.0.2	Perbaikan bug dan peningkatan sinkronisasi prioritas aset dengan edge.

Versi	Perubahan
2.0.1	Versi awal.

Lihat juga

- Apa itu AWS IoT SiteWise? dalam AWS IoT SiteWise User Guide.

Simulator sumber data IoT SiteWise OPC-UA

Komponen simulator sumber data IoT SiteWise OPC-UA (`aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator`) memulai server OPC-UA lokal yang menghasilkan data sampel. Gunakan server OPC-UA ini untuk mensimulasikan sumber data yang dibaca oleh komponen kolektor IoT SiteWise OPC-UA pada gateway. AWS IoT SiteWise Kemudian, Anda dapat menjelajahi AWS IoT SiteWise fitur menggunakan data sampel ini. Untuk informasi selengkapnya tentang AWS IoT SiteWise gateway, lihat Menggunakan AWS IoT SiteWise di tepi di AWS IoT SiteWisePanduan Pengguna.

Topik

Versi

Komponen ini memiliki versi berikut:

- 1.0.x

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Greengrass harus dapat menggunakan port 4840 pada host lokal. Server OPC-UA lokal komponen ini berjalan di port ini.

Dependensi

Saat Anda men-deploy komponen, AWS IoT Greengrass juga men-deploy versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

Tabel berikut mencantumkan dependensi untuk semua versi komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.3.0 <3.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/
  aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs
  \aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator.log -Tail 10 -Wait
  ```

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
1.0.0	Versi awal.

Versi	Perubahan
	Menambahkan dukungan untuk Windows Server 2016 atau lebih tinggi.

Lihat juga

- Apa ituAWS IoT SiteWise? dalam AWS IoT SiteWiseUser Guide.

Penerbit IoT SiteWise

Komponen SiteWise penerbit IoT (`aws.iot.SiteWiseEdgePublisher`) memungkinkan AWS IoT SiteWise gateway untuk mengekspor data dari tepi ke. AWS Cloud

Untuk informasi selengkapnya tentang AWS IoT SiteWise gateway, lihat Menggunakan AWS IoT SiteWise di tepi di AWS IoT SiteWise Panduan Pengguna.

Topik

Versi

Komponen ini memiliki versi berikut:

- 3.1.x
- 3.0.x
- 2.4.x
- 2.3.x
- 2.2.x
- 2.1.x
- 2.0.x

Jenis

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Greengrass harus berjalan di salah satu platform berikut:
  - OS: Ubuntu 18.04 atau yang lebih baru

    Arsitektur: x86_64 (AMD64) atau ARMv8 (Aarch64)
  - OS: Perusahaan Red Hat Linux (RHEL) 8

    Arsitektur: x86_64 (AMD64) atau ARMv8 (Aarch64)
  - OS: Amazon Linux 2

    Arsitektur: x86_64 (AMD64) atau ARMv8 (Aarch64)

- OS: Debian 11

  Arsitektur: x86_64 (AMD64) atau ARMv8 (Aarch64)

- OS: Windows Server 2019 atau yang lebih baru

  Arsitektur: x86_64 (AMD64)

- Perangkat inti Greengrass harus terhubung ke Internet.
- Perangkat inti Greengrass harus diberi wewenang untuk melakukan tindakan. `iotsitewise:BatchPutAssetPropertyValue` Untuk informasi selengkapnya, lihat Mengotorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

  Example kebijakan izin

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

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Wajib	Deskripsi
`data.iotsitewise.` *region*`.amazonaws.com`	443	Ya	Publikasikan data ke AWS IoT SiteWise.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

Tabel berikut mencantumkan dependensi untuk versi 2.0.x hingga 2.2.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Inti Greengrass	>=2.3.0<3.0.0	Keras
Manajer pengaliran	>=2.0.10 <3.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

Anda dapat menggunakan AWS IoT SiteWise konsol atau API untuk mengonfigurasi komponen SiteWise penerbit IoT. Untuk informasi selengkapnya, lihat Langkah 3: Mengkonfigurasi penerbit - opsional di Panduan AWS IoT SiteWise Pengguna.

Data input

Komponen ini membaca `PutAssetPropertyValueEntry` pesan dari manajer AWS IoT Greengrass aliran. Untuk informasi selengkapnya, lihat PutAssetPropertyValueEntrydi Referensi AWS IoT SiteWise API.

Berkas log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log -Tail 10 -
  Wait
  ```

Pemecahan masalah dan debugging

Komponen ini mencakup log peristiwa baru untuk membantu pelanggan mengidentifikasi dan memperbaiki masalah. File log terpisah dari file log lokal, dan ditemukan di lokasi berikut. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/logs/
IotSiteWisePublisherEvents.log
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgePublisher\logs
\IotSiteWisePublisherEvents.log
```

Log ini mencakup informasi terperinci dan instruksi pemecahan masalah. Informasi pemecahan masalah disediakan bersama diagnostik, dengan deskripsi tentang cara memperbaiki masalah, dan terkadang dengan tautan ke informasi lebih lanjut. Informasi diagnostik meliputi:

- Tingkat kepelikan
- Stempel Waktu
- Informasi khusus acara tambahan

Example Contoh log

```
accountBeingThrottled:
  Summary: Data upload speed slowed due to quota limits
  Level: WARN
  Timestamp: '2023-06-09T21:30:24.654Z'
  Description: The IoT SiteWise Publisher is limited to the "Rate of data points
 ingested"
    quota for a customers account. See the associated documentation and associated
    metric for the number of requests that were limited for more information. Note
    that this may be temporary and not require any change, although if the issue
 continues
    you may need to request an increase for the mentioned quota.
  FurtherInformation:
  - https://docs.aws.amazon.com/iot-sitewise/latest/userguide/quotas.html
  - https://docs.aws.amazon.com/iot-sitewise/latest/userguide/troubleshooting-
gateway.html#gateway-issue-data-streams
  AssociatedMetrics:
  - Name: TotalErrorCount
    Description: The total number of errors of this type that occurred.
    Value: 327724.0
  AssociatedData:
  - Name: AggregatePropertyAliases
    Description: The aggregated property aliases of the throttled data.
    FileLocation: /greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/./logs/data/
AggregatePropertyAliases_1686346224654.log
```

Lisensi

Komponen ini dirilis menurut Perjanjian Lisensi Perangkat Lunak Greengrass Core.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan

3.1.3	Perbaikan bug dan peningkatan
	• Menyelesaikan masalah di mana file log peristiwa berada di `/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/logs/IoTSiteWisePublisherEvents.log` dibuat tetapi tidak ada peristiwa yang dicatat.
	• Menambahkan CloudWatch metrik berikut untuk memantau koneksi dengan broker MQTT:
	  • `IoTSiteWisePublisher.IsConnectedToMqttBroker`
	  • `IoTSiteWisePublisher.NumberOfSubscriptionsToMqttBroker`
	  • `IoTSiteWisePublisher.NumberOfUniqueMqttTopicsReceived`
	  • `IoTSiteWisePublisher.MqttMessageReceivedSuccessCount`
	  • `IoTSiteWisePublisher.MqttReceivedSuccessBytes`
	  Untuk informasi selengkapnya tentang metrik ini, lihat metrik AWS IoT Greengrass Version 2 gateway.
	• Menyelesaikan masalah di mana `BatchCreateJob` API masih akan dipanggil meskipun mengunggah file parket ke S3 gagal.

3.1.2	Perbaikan bug dan peningkatan • Memperbaiki masalah penggunaan CPU tinggi yang diperkenalkan di versi 3.1.1.
3.1.1	Perbaikan bug dan peningkatan • Menambahkan logging tambahan yang mengidentifikasi alias data yang terpengaruh ketika terjadi kesalahan. • Menambahkan penegakan lokal batas AWS IoT SiteWise API pada usia data yang dicerna.

Versi	Perubahan
	• Memperbaiki masalah saat Publisher mencampur pos pemeriksaan StreamManager aliran saat ada beberapa tujuan Amazon S3. • Memperbaiki hambatan kinerja dengan cara Penerbit membaca dari aliran. StreamManager
3.1.0	Fitur baru • Menambahkan dukungan untuk mempublikasikan data sebagai file parket ke Amazon S3. • Menambahkan dukungan untuk konsumsi AWS IoT SiteWise buffer.
3.0.0	Perbaikan bug dan peningkatan • Memperbaiki masalah yang terkait dengan dukungan proxy. Fitur baru • Memungkinkan dukungan konsumsi data dari broker MQTT.
2.4.1	Perbaikan bug dan peningkatan • Aktifkan komponen untuk bekerja dengan Java Corretto 11 versi 11.0.20.8.1 dan yang lebih baru. Versi komponen 2.4.0 dan 2.3.3 menampilkan pesan `"Could not find or load main class"` kesalahan saat digunakan dengan Java Corretto versi 11.0.20.8.1.
2.4.0	Fitur baru • Menambahkan log peristiwa baru untuk membuatnya lebih mudah untuk mengidentifikasi dan memperbaiki masalah. Perbaikan bug dan peningkatan • Meningkatkan pemulihan pos pemeriksaan Publisher.
2.3.3	Perbaikan bug dan peningkatan • Meningkatkan kemampuan untuk mendukung throughput tinggi.
2.3.2	Perbaikan bug dan peningkatan • Memperbaiki dukungan proxy HTTP saat mengunduh konfigurasi Penerbit.

Versi	Perubahan

2.3.1	Fitur baru
	• Menambahkan dukungan untuk menginstal Data Collection Pack pada arsitektur Linux ARMv8.
	• Persyaratan minimum untuk Linux ARMv8:
	  • Memori: 4 GB
	  • CPU: ARM Cortex-A72 atau spesifikasi yang setara

2.2.3	Perbaikan bug dan peningkatan • Menghapus coba lagi untuk pengecualian generik yang tidak ada dalam daftar pengecualian yang dapat diambil.
2.2.2	Perbaikan bug dan peningkatan • Memperkenalkan kembali dukungan unggahan data ke AWS IoT SiteWise melalui server proxy HTTP.
2.2.1	**Note** Versi ini tidak mendukung konfigurasi proxy HTTP. Versi 2.2.2 dan yang lebih tinggi memperkenalkan kembali dukungan untuk fitur ini. Fitur baru • Menambahkan dukungan ke komponen ini untuk mengaktifkan kompresi saat mengunggah data ke. AWS IoT SiteWise

Versi	Perubahan

2.2.0	**Note**
	Versi ini tidak mendukung konfigurasi proxy HTTP. Versi 2.2.2 dan yang lebih tinggi memperkenalkan kembali dukungan untuk fitur ini.
	
	Fitur baru
	• Memperbarui komponen ini untuk mengompres data sebelum mengirimnya ke AWS IoT SiteWise layanan.
	  • Dalam kebanyakan kasus, perubahan ini mengurangi penggunaan bandwidth hingga 75 persen dibandingkan dengan versi sebelumnya dari komponen ini.
	  • Dalam kebanyakan kasus, perubahan ini meningkatkan penggunaan CPU hingga 5 persen. Pada gateway yang memproses data dalam jumlah besar, perubahan ini dapat meningkatkan penggunaan CPU hingga 15 persen.
	  • Perubahan ini tidak memengaruhi biaya AWS IoT SiteWise layanan atau penggunaan kuota layanan.
	  • Menambahkan dukungan untuk Windows Server 2019 atau lebih tinggi.
	Perbaikan bug dan peningkatan
	• Memperbaiki masalah yang mencegah komponen ini dimulai saat file pos pemeriksaan rusak.

2.1.4	Perbaikan bug dan peningkatan • Memperbaiki kompatibilitas dengan Java versi 8.

Versi	Perubahan
2.1.3	**Warning** Versi ini tidak lagi tersedia, kecuali di Wilayah AS Timur (Ohio), Kanada (Tengah), dan AWS GovCloud (AS-Timur). Versi komponen ini membutuhkan Java versi 11 atau lebih untuk dijalankan. Perbaikan dalam versi ini tersedia di versi yang lebih baru dari komponen ini. Perbaikan bug dan peningkatan • Memperbaiki pesan galat saat Anda menerapkan komponen ini pada perangkat yang tidak didukung. • Pembaruan untuk mencatat kesalahan saat pengunggahan data gagal.
2.1.2	Perbaikan bug dan peningkatan • Pembaruan untuk memanggil fitur ekspor data yang kedaluwarsa segera setelah data kedaluwarsa.
2.1.1	Perbaikan bug dan perbaikan.
2.1.0	Fitur baru • Menambahkan dukungan untuk mempublikasikan data terbaru ke cloud terlebih dahulu. • Menambahkan dukungan untuk tidak menerbitkan data kedaluwarsa ke cloud. • Menambahkan dukungan untuk menyimpan data kedaluwarsa secara lokal. Perbaikan bug dan peningkatan • Mengurangi disk I/O dan latensi yang sesuai.
2.0.2	Perbaikan bug dan perbaikan.
2.0.1	Versi awal.

Lihat juga

- Apa itu AWS IoT SiteWise? dalam AWS IoT SiteWise User Guide.

Prosesor IoT SiteWise

Komponen SiteWise prosesor IoT (`aws.iot.SiteWiseEdgeProcessor`) memungkinkan AWS IoT SiteWise gateway untuk memproses data di tepi.

Dengan komponen ini, AWS IoT SiteWise gateway dapat menggunakan model aset dan aset untuk memproses data pada perangkat gateway. Untuk informasi selengkapnya tentang AWS IoT SiteWise gateway, lihat Menggunakan AWS IoT SiteWise di tepi di AWS IoT SiteWise Panduan Pengguna.

Topik

Versi

Komponen ini memiliki versi berikut:

- 3.2.x
- 3.1.x
- 3.0.x
- 2.2.x
- 2.1.x

- 2.0.x

Tipe

Komponen ini adalah komponen generik (`aws.greengrass.generic`). Inti Greengrass menjalankan skrip siklus hidup komponen.

Untuk informasi selengkapnya, lihat Jenis komponen.

Sistem operasi

Komponen ini dapat diinstal pada perangkat inti yang menjalankan sistem operasi berikut:

- Linux
- Windows

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Perangkat inti Greengrass harus berjalan di salah satu platform berikut:
  - OS: Ubuntu 20.04 atau 18.04

    Arsitektur: x86_64 (AMD64)
  - OS: Perusahaan Red Hat Linux (RHEL) 8

    Arsitektur: x86_64 (AMD64)
  - OS: Amazon Linux 2

    Arsitektur: x86_64 (AMD64)
  - OS: Windows Server 2019 atau yang lebih baru

    Arsitektur: x86_64 (AMD64)
- Perangkat inti Greengrass harus memungkinkan lalu lintas masuk di port 443.
- Perangkat inti Greengrass harus memungkinkan lalu lintas keluar di port 443 dan 8883.
- Port berikut dicadangkan untuk digunakan oleh AWS IoT SiteWise: 80, 443, 3001, 4569, 4572, 8000, 8081, 8082, 8084, 8085, 8086, 8445, 9000, 9500, 11080, dan 50010. Menggunakan port cadangan untuk lalu lintas dapat mengakibatkan koneksi terputus.

> **Note**
>
> Port 8087 hanya diperlukan untuk versi 2.0.15 dan yang lebih baru dari komponen ini.

- Peran perangkat Greengrass harus memiliki izin yang memungkinkan Anda menggunakan gateway di perangkat Anda. AWS IoT SiteWise AWS IoT Greengrass V2 Untuk informasi selengkapnya, lihat Persyaratan dalam Panduan AWS IoT SiteWise Pengguna.

Titik akhir dan port

Komponen ini harus dapat melakukan permintaan keluar ke titik akhir dan port berikut, selain titik akhir dan port yang diperlukan untuk operasi dasar. Untuk informasi selengkapnya, lihat Izinkan lalu lintas perangkat melalui proxy atau firewall.

Titik Akhir	Port	Wajib	Deskripsi
`model.iotsitewise.region.amazonaws.com`	443	Ya	Dapatkan informasi tentang AWS IoT SiteWise aset dan model aset Anda.
`edge.iotsitewise.region.amazonaws.com`	443	Ya	Dapatkan informasi tentang konfigurasi AWS IoT SiteWise gateway perangkat inti.
`ecr.region.amazonaws.com`	443	Ya	Unduh gambar

Titik Akhir	Port	Wajib	Deskripsi
			AWS IoT SiteWise Edge gateway Docker dari Amazon Elastic Container Registry.
`iot.region.amazonaws.com`	443	Ya	Dapatkan titik akhir perangkat untuk Anda Akun AWS.
`sts.region.amazonaws.com`	443	Ya	Dapatkan ID Anda Akun AWS.
`monitor.iotsitewise. region.amazonaws.com`	443	Tidak	Diperlukan jika Anda mengakses AWS IoT SiteWise Monitor portal pada perangkat inti.

Dependensi

Saat Anda menerapkan komponen, gunakan AWS IoT Greengrass juga versi dependensinya yang kompatibel. Ini berarti bahwa Anda harus memenuhi persyaratan untuk komponen dan semua dependensinya untuk berhasil men-deploy komponen. Bagian ini berisi daftar dependensi untuk versi

yang dirilis dari komponen ini dan kendala versi semantik yang menentukan versi komponen untuk setiap dependensi. Anda juga dapat melihat dependensi untuk setiap versi komponen di konsol AWS IoT Greengrass tersebut. Pada halaman detail komponen, cari daftar Dependensi.

Tabel berikut mencantumkan dependensi untuk versi 2.0.x hingga 2.1.x komponen ini.

Dependensi	Versi yang kompatibel	Jenis dependensi
Layanan pertukaran Token	>=2.0.3 <3.0.0	Keras
Manajer pengaliran	>=2.0.10 <3.0.0	Keras
CLI Greengrass	>=2.3.0 <3.0.0	Keras

Untuk informasi selengkapnya tentang dependensi komponen, lihat referensi resep komponen.

Konfigurasi

Komponen ini tidak memiliki parameter konfigurasi apapun.

File log lokal

Komponen ini menggunakan file log berikut.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeProcessor.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeProcessor.log
```

Untuk melihat log komponen ini

- Jalankan perintah berikut pada perangkat inti untuk melihat file log komponen ini secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan jalur ke folder AWS IoT Greengrass root.

Linux

```
sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgeProcessor.log
```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeProcessor.log -Tail 10 -
Wait
```

Lisensi

Komponen ini mencakup perangkat lunak/lisensi pihak ketiga berikut:

- Apache-2.0
- MIT
- BSD-2-klausul
- BSD-3-klausul
- CDDL-1.0
- CDDL-1.1
- ISC
- Zlib
- GPL-3.0-dengan-GCC-pengecualian
- Domain Publik
- Python-2.0
- Unicode-DFS-2015
- BSD-1-klausul
- OpenSSL
- EPL-1.0
- EPL-2.0
- GPL-2.0-dengan-Classpath-Exception
- MPL-2.0
- CC0-1.0

- JSON

Komponen ini dirilis menurut Perjanjian Lisensi Perangkat Lunak Greengrass Core.

Changelog

Tabel berikut menjelaskan perubahan dalam setiap versi komponen.

Versi	Perubahan
3.2.1	Perbaikan bug dan peningkatan • Perbaiki masalah di mana panggilan AWS IoT SiteWise API tidak melakukan paginasi secara sinkron dengan SiteWise Edge. • Perbaiki masalah untuk tidak mempublikasikan `MessageRemaining.SiteWise_Edge_Stream` metrik lagi. • Menambahkan CloudWatch metrik berikut untuk memantau koneksi dengan broker MQTT. • `IoTSiteWiseProcessor.IsConnectedToMqttBroker` • `IoTSiteWiseProcessor.NumberOfSubscriptionsToMqttBroker` • `IoTSiteWiseProcessor.NumberOfUniqueMqttTopicsReceived` • `IoTSiteWiseProcessor.MqttMessageReceivedSuccessCount` • `IoTSiteWiseProcessor.MqttReceivedSuccessBytes` Untuk informasi selengkapnya tentang metrik ini, lihat metrik AWS IoT Greengrass Version 2 gateway.
3.2.0	Peningkatan Performa, dll.. • Optimalkan layanan API agar memiliki jejak memori yang lebih kecil dan membutuhkan lebih sedikit ruang disk untuk menginstal • Ini memberikan pengurangan 2 GB dalam penggunaan memori awal (sekarang menggunakan 7,5 GB memori saat startup, namun 16 GB masih disarankan) dan pengurangan ukuran unduhan 500 MB (sekarang memerlukan unduhan 1,4 GB) untuk seluruh komponen.

Versi	Perubahan
	Fitur baru • `GetAssetPropertyValueAggregates` API sekarang mendukung jendela agregasi 15 menit di tepi. • Port 8081 dan 8082 tidak perlu lagi tersedia agar komponen ini berjalan dengan benar. **Note** Titik akhir lokal untuk API bidang AWS IoT SiteWise data, seperti`get-asset-property-value` , sedang diubah dari `http://localhost:8081` kehttp://localhost:11080/data . Titik akhir lokal untuk API bidang AWS IoT SiteWise kontrol, seperti`list-asset-models` , sedang diubah dari `http://localhost:11080` kehttp://localhost:11080/control . AWS selalu merekomendasikan agar Anda menggunakan titik akhir HTTPS gateway SiteWise Edge. Titik akhir tersebut tidak berubah. Perbaikan bug dan peningkatan • Sinkronisasi dari sekarang AWS IoT SiteWise akan mentransisikan sumber daya ke status valid jika sinkronisasi sebelumnya terputus. Ini akan memperbaiki masalah dengan beberapa sumber daya yang rusak setelah restart paksa. • Memperbaiki kondisi langka di mana sumber daya dapat rusak di tepi jika dimodifikasi selama sinkronisasi. Sinkronisasi sekarang akan gagal jika kondisi ini terdeteksi, dan sumber daya akan dicoba lagi pada sinkronisasi berikutnya. • Memperbaiki masalah yang memungkinkan titik akhir HTTP untuk API dipanggil secara eksternal. Hanya HTTPS yang dapat digunakan untuk memanggil API di luar alamat loopback lokal sekarang. • `ListAssets` API sekarang menunjukkan hierarki aset untuk aset yang disimpan di edge. • Memperbaiki masalah saat Paket Pemrosesan Data gagal memulai ulang, meningkatkan, atau menurunkan versi pada Windows.

Versi	Perubahan
	• Memperbaiki bug dalam Paket Pemrosesan Data untuk OS Windows yang mencegah pelanggan menggunakan kredensi untuk terhubung dengan Broker MQTT.
3.1.3	Perbaikan bug dan peningkatan • Perbaiki masalah di mana Paket Pemrosesan Data salah melaporkan sinkronisasi yang berhasil ketika beberapa sumber daya benar-benar gagal. • Izinkan beberapa aset memiliki nama yang sama selama mereka tidak memiliki induk yang sama.
3.1.1	Perbaikan bug dan peningkatan • Perbaiki masalah di mana permintaan SigV4 gagal karena ketidakco cokan zona waktu. • Perbaiki masalah di mana properti transformasi dan metrik berhenti menghitung ketika mereka mengandalkan atribut setelah memulai ulang. • Aktifkan dukungan konfigurasi Port Stream Manager kustom. • Perbaiki masalah di mana properti yang disinkronkan ke tepi mungkin berhenti diperbarui.
3.1.0	Perbaikan bug dan peningkatan • Perbaiki masalah di mana `ListAssetModels` API gagal menghasil kan token berikutnya.
3.0.0	Fitur baru • Memungkinkan dukungan konsumsi data dari broker MQTT.

Versi	Perubahan
2.2.1	Perbaikan bug dan peningkatan • Sesuaikan proses sinkronisasi untuk membuat penyimpanan data bidang kontrol lebih konsisten dengan cara cloud beroperasi. Ini sedikit berdampak pada peningkatan. **Note** Data bidang kontrol yang disinkronkan pada versi 2.2.1 atau lebih tinggi tidak akan kompatibel dengan versi sebelumnya. Untuk menurunkan versi ke versi sebelumnya, Anda harus menyelesaikan instalasi baru. Ini tidak memengaruhi peningkat an, data yang disinkronkan pada versi sebelumnya akan berfungsi dengan versi 2.2.1. • Modifikasi tambahan pada rantai AWS kredensional untuk mempriori taskan AWS IoT Greengrass V2 kredensional.
2.1.37	Perbaikan bug dan peningkatan • Menghentikan dependency-routing-service proses dan memindahk an fungsionalitasnya ke dalam property-state-service proses untuk mengurangi penggunaan sumber daya dari proses komunikasi. • Tingkatkan batas hasil maksimum untuk `get-asset-property-value-history` API menjadi 20.000 agar sesuai dengan batas yang digunakan oleh AWS IoT SiteWise. • Perbaiki masalah saat token berikutnya tidak disediakan dalam hasil paginasi untuk `get-asset-property-value-history` API saat tidak ada batas hasil maksimal yang ditentukan.
2.1.35	Perbaikan bug dan peningkatan • Memodifikasi rantai AWS kredensil untuk memprioritaskan kredensil. AWS IoT Greengrass • Memperbaiki masalah dengan deteksi akun saat menerapkan sebagai bagian dari grup AWS IoT Thing.

Versi	Perubahan
2.1.34	Perbaikan bug dan peningkatan • Menyesuaikan perhitungan metrik/transformasi untuk menggunakan multi-threading di Linux. Windows terus menjalankan perhitungan single-threaded untuk kompatibilitas. • Memperbaiki masalah di mana perhitungan metrik akan hilang untuk beberapa jendela komputasi.
2.1.33	Perbaikan bug dan peningkatan • Memperbaiki masalah dengan pelaporan status kesalahan ke konsol Greengrass.
2.1.32	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk nama pengguna dan grup yang disesuaikan.
2.1.31	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk menghitung rata-rata tertimbang waktu dan standar deviasi tertimbang waktu untuk data yang dimodelkan. AWS IoT SiteWise
2.1.29	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk memfilter aset pada fungsionalitas edge.
2.1.28	Perbaikan bug dan peningkatan • Mengoptimalkan sinkronisasi sumber daya untuk memungkinkan sejumlah besar aset disinkronkan dari tepi AWS Cloud ke tepi.
2.1.24	Perbaikan bug dan peningkatan • Memperbaiki masalah yang menyebabkan dasbor menghilang saat menyinkronkan sumber daya untuk kedua kalinya.

Versi	Perubahan
2.1.23	Perbaikan bug dan peningkatan • Menambahkan batas waktu untuk proses `aws.iot.SiteWiseEdgeProcessor` instalasi untuk menghindari kegagalan instalasi jika konektivitas internet lambat. • Sinkronisasi sumber daya yang dioptimalkan untuk meningkatkan efisiensi sinkronisasi antara cloud dan edge.
2.1.21	**Warning** Upgrade dari 2.0.x ke 2.1.x akan mengakibatkan hilangnya data lokal. Fitur baru • Menambahkan dukungan untuk Windows Server 2019 atau lebih tinggi. • Menghapus docker untuk sistem operasi berbasis Linux.
2.0.16	Versi ini berisi perbaikan bug dan perbaikan.
2.0.15	Perbaikan bug dan peningkatan • Mengubah port yang digunakan komponen ini untuk operasi API sinkronisasi sumber daya dari 8085 menjadi 8087. Akibatnya, komponen ini sekarang membutuhkan port 8087 untuk tersedia. Komponen ini masih membutuhkan port 8085 untuk tersedia. • Memperbarui AWS OpsHub otentikasi untuk menolak pengguna yang tidak sah selama login, bukan saat pengguna mencoba memanggil operasi API.
2.0.14	Versi ini berisi perbaikan bug dan perbaikan.
2.0.13	Perbaikan bug dan peningkatan • Memperbaiki masalah sehingga ketika komponen ini melaporkan data ke CloudWatch metrik Amazon, sekarang dengan benar menunjukkan data mana yang tidak dimodelkan.

Versi	Perubahan
2.0.9	Perbaikan bug dan peningkatan • Meningkatkan keandalan untuk membuat dan memperbarui AWS IoT SiteWise sumber daya pada perangkat inti. • Menambahkan operasi API lokal tambahan yang dapat Anda gunakan untuk memantau komponen mana yang diinstal pada perangkat inti, versi setiap komponen, dan status setiap komponen. Anda dapat melihat informasi ini di tab Pengaturan di AWS IoT SiteWise aplikasi AWS OpsHub for pada perangkat inti. • Menambahkan status kesehatan untuk kontainer Docker yang dijalanka n komponen ini. Anda dapat menjalankan `docker ps` perintah untuk melihat status kesehatan kontainer.
2.0.7	Perbaikan bug dan peningkatan • Memperbaiki dukungan untuk melihat AWS IoT SiteWise Monitor portal pada perangkat inti.
2.0.6	Perbaikan bug dan peningkatan • Memperbaiki AWS IoT SiteWise `statetime()` ,`earliest()` , dan `latest()` fungsi yang dihitung komponen ini pada perangkat inti.
2.0.5	Perbaikan bug dan peningkatan • Menambahkan dukungan untuk AWS IoT SiteWise `pretrigger()` fungsi dalam transformasi yang dihitung komponen ini pada perangkat inti. • Mengubah jalur tempat komponen ini menyimpan konfigurasi Lightweig ht Directory Access Protocol (LDAP) untuk otentikasi.
2.0.2	Versi awal.

Lihat juga

- Apa itu AWS IoT SiteWise? dalam AWS IoT SiteWise User Guide.

Komponen yang didukung penerbit

Komponen yang didukung penerbit berada dalam rilis pratinjau untuk AWS IoT Greengrass dan dapat berubah sewaktu-waktu. Komponen-komponen ini tidak didukung olehAWS. Anda harus menghubungi Penerbit untuk masalah apa pun dengan masing-masing komponen.

Komponen yang didukung Greengrass Publisher dikembangkan, ditawarkan, dan dilayani oleh vendor komponen pihak ketiga. Vendor komponen pihak ketiga berasal dari Katalog AWS Partner Perangkat, AWS Pahlawan, atau vendor komunitas. Anda dapat membeli komponen dalam katalog ini dengan menghubungi vendor komponen pihak ketiga secara langsung.

Komponen yang didukung Greengrass Publisher meliputi yang berikut ini:

Topik

Aishield.edge

Komponen ini dikembangkan dan didukung oleh AiShield, didukung oleh Bosch. Tingkatkan keamanan AI Anda dengan Aishield.edge. Komponen ini dirancang untuk menyebarkan pertahanan yang diinformasikan ancaman dan disesuaikan dengan mulus ke perangkat edge, yang melindungi perangkat Anda dari serangan AI.

Komponen ini menawarkan manfaat sebagai berikut:

- Transisi mulus dari analisis kerentanan dengan AiShield AI Security ke pertahanan tepi yang diperkaya di dalamnya AWS
- Terapkan pertahanan yang disesuaikan di beberapa perangkat tepi dengan mudah
- Perlindungan luas yang disesuaikan dengan beragam pengaturan AI yang mendukung berbagai jenis model dan kerangka kerja
- Tetap diperbarui dengan integrasi yang mulus ke dalam Amazon SageMaker dan alur kerja Greengrass

- Dapatkan wawasan langsung tentang potensi ancaman, dengan data yang disampaikan langsung ke AWS IoT Core
- Jalur keamanan AI yang kohesif untuk penyebaran pertahanan di tepi dari AiShield AI Security di Marketplace AWS

Komponen ini harus berjalan pada platform berikut:

- OS: Linux

Jika Anda tertarik untuk membeli komponen ini, hubungi Perangkat Lunak Bosch dan Solusi Digital: <AIShield.Contact@bosch.com>.

EdgeLabs Sensor AI

Komponen ini dikembangkan dan didukung oleh AI EdgeLabs. AI EdgeLabs Sensor adalah aplikasi berbasis wadah yang berisi kemampuan deteksi dan pencegahan ancaman berbasis AI. Sensor AI dibungkus menjadi komponen Greengrass dan digunakan sebagai wadah mandiri pada perangkat inti bersama komponen Greengrass lainnya.

Komponen saat ini adalah agen berbasis kontainer yang terus memverifikasi komunikasi jaringan, mencari pola ancaman dalam perangkat lunak yang berjalan di Edge Host atau gateway IoT. Komponen ini menggunakan eBPF, verifikasi perilaku bandwidth proses, dan konfigurasi berbasis host. Fungsionalitas utama komponen ini didasarkan pada fungsi NDR/IPS dan EDR.

Komponen ini menawarkan manfaat sebagai berikut:

- Deteksi ancaman berbasis AI terhadap serangan jaringan dan malware (EDR/NDR)
- Respons Insiden Berbasis AI Otomatis (IPS)
- Intelijen ancaman host-lokal dengan transfer data minimal di luar
- Penerapan ringan dengan Docker dan Greengrass

Komponen ini harus berjalan di salah satu platform berikut:

- OS: Linux

Jika Anda tertarik untuk membeli komponen ini, hubungi AI EdgeLabs: <contact@edgelabs.ai>.

Greengrass S3 Ingestor

Komponen ini dikembangkan dan didukung oleh Nathan Glover. Komponen Greengrass S3 Ingestor dirancang untuk digunakan dengan komponen manajer aliran. Komponen ini mengambil aliran pesan JSON yang dibatasi baris dari pengelola aliran dan mengumpulkannya ke dalam file GZIP. Komponen ini memungkinkan konsumsi data yang efisien ke Amazon S3 untuk diproses lebih lanjut atau untuk penyimpanan. Komponen ini tidak mendukung pengiriman data ke AWS Cloud dalam waktu nyata.

Komponen ini harus berjalan di salah satu platform berikut:

- OS: Linux
- OS: Windows

```
<Jika Anda tertarik untuk membeli komponen ini, hubungi Nathan Glover: nathan@gl
```

Komponen komunitas

Katalog Perangkat Lunak Greengrass adalah indeks komponen Greengrass yang dikembangkan oleh komunitas Greengrass. Dari katalog ini, Anda dapat mengunduh, memodifikasi, dan menyebarkan komponen untuk membuat aplikasi Greengrass Anda. Anda dapat melihat katalog di tautan berikut: https://github.com/aws-greengrass/ aws-greengrass-software-catalog.

Setiap komponen memiliki GitHub repositori publik yang dapat Anda jelajahi. Lihat Katalog Perangkat Lunak Greengrass untuk menemukan daftar GitHub lengkap komponen komunitas. Misalnya, katalog ini mencakup komponen-komponen berikut:

- Aliran Video Amazon Kinesis

  Komponen ini menyerap aliran audio dan video dari kamera lokal yang menggunakan Real Time Streaming Protocol (RTSP). Komponen kemudian mengunggah streaming audio dan video ke Amazon Kinesis Video Streams.
- Gerbang IoT Bluetooth

  Komponen ini menggunakan BluePypustaka yang memungkinkan komunikasi dengan perangkat Bluetooth Low Energy (LE) untuk membuat antarmuka klien Bluetooth LE.
- Sertifikat Rotator

Komponen ini menyediakan sarana untuk memutar sertifikat perangkat AWS IoT Greengrass inti dan kunci pribadi, di seluruh armada Anda, dalam skala besar.

- Terowongan aman dalam peti kemas

  Komponen ini menyediakan wadah Docker untuk tunneling aman dengan semua dependensi dan pustaka yang cocok dalam resep yang dapat digunakan kembali yang tidak bergantung pada sistem operasi host tertentu.

- Grafana

  Komponen ini memungkinkan Anda untuk meng-host server Grafana pada perangkat inti Greengrass. Anda dapat menggunakan dasbor Grafana untuk memvisualisasikan dan mengelola data pada perangkat inti.

- GStreamer untuk Amazon Lookout for Vision

  Komponen ini menyediakan plugin GStreamer sehingga Anda dapat melakukan deteksi anomali Lookout for Vision di pipeline GStreamer kustom Anda.

- Asisten rumah

  Komponen ini memungkinkan pelanggan untuk menggunakan Home Assistant untuk menyediakan kontrol lokal perangkat rumah pintar. Ini menyediakan integrasi dengan AWS layanan di edge dan di cloud untuk memberikan solusi otomatisasi rumah yang memperluas Home Assistant.

- Dasbor InfluxDBGrafana

  Komponen ini memberikan pengalaman sekali klik untuk menyiapkan komponen InfluxDB dan Grafana. Ini menghubungkan InfluxDB ke Grafana dan mengotomatiskan pengaturan dasbor Grafana lokal yang membuat telemetri secara real time. AWS IoT Greengrass

- InfluxDB

  Komponen ini menyediakan database deret waktu InfluxDB pada perangkat inti Greengrass. Anda dapat menggunakan komponen ini untuk memproses data dari sensor IoT, menganalisis data secara real time, dan memantau operasi di edge.

- Penerbit InfluxDB

  Komponen ini menyampaikan telemetri kesehatan AWS IoT Greengrass sistem dari plugin pemancar Nucleus ke InfluxDB. Komponen ini juga dapat meneruskan telemetri khusus ke InfluxDB.

- Kerangka kerja pubsub IoT

Kerangka kerja ini menyediakan arsitektur aplikasi, kode template, dan contoh yang dapat diterapkan yang membantu meningkatkan kualitas kode untuk aplikasi pubsub IoT berbasis peristiwa terdistribusi menggunakan komponen kustom v2. AWS IoT Greengrass Untuk informasi selengkapnya, lihat Buat AWS IoT Greengrass komponen.

- Lab Jupyter

  Komponen ini menyebar JupyterLab ke perangkat AWS IoT Greengrass inti. Lingkungan Jupyter memiliki akses ke sumber daya variabel proses dan lingkungan yang ditetapkan oleh AWS IoT Greengrass, menyederhanakan proses pengujian dan pengembangan komponen yang ditulis dengan Python.

- Server web lokal

  Komponen ini memungkinkan Anda untuk membuat antarmuka pengguna web lokal pada perangkat inti Greengrass. Anda dapat membuat antarmuka pengguna web lokal yang memungkinkan Anda mengonfigurasi pengaturan perangkat dan aplikasi atau memantau perangkat, misalnya.

- LoRaWaAdaptor protokol N

  Komponen ini menyerap data dari perangkat nirkabel lokal yang menggunakan protokol LoRaWa N, yang merupakan protokol jaringan area lebar daya rendah (LPWAN). Komponen ini memungkinkan Anda untuk menganalisis dan bertindak pada data secara lokal tanpa berkomunikasi dengan cloud.

- Modbus TCP

  Komponen ini mengumpulkan data dari perangkat lokal menggunakan protokol ModBustCP dan menerbitkannya ke aliran data yang dipilih.

- Simpul-merah

  Komponen ini menginstal Node-Red pada perangkat AWS IoT Greengrass inti menggunakan NPM. Komponen bergantung pada komponen Node-Red Auth yang harus digunakan dan dikonfigurasi secara eksplisit. Anda dapat menggunakan CLI Node-Red untuk Greengrass untuk menyebarkan aliran Node-Red ke perangkat. AWS IoT Greengrass

- Docker merah simpul

  Komponen ini menginstal Node-Red pada perangkat AWS IoT Greengrass inti menggunakan wadah Node-Red Docker resmi. Komponen bergantung pada komponen Node-Red Auth yang

harus digunakan dan dikonfigurasi secara eksplisit. Anda dapat menggunakan CLI Node-Red untuk Greengrass untuk menyebarkan aliran Node-Red ke perangkat. AWS IoT Greengrass

- Auth Node-merah

  Komponen ini mengonfigurasi nama pengguna dan kata sandi untuk mengamankan instance Node-Red yang berjalan pada perangkat inti. AWS IoT Greengrass

- OpenThreadRouter Perbatasan

  Komponen ini menyebarkan wadah OpenThread Border Router Docker. Komponen ini membantu menyusun perangkat Matter yang menyertakan router perbatasan Thread.

- Konektor Data Streaming OSI Pi

  Komponen ini menyediakan streaming konsumsi data real-time dari OSI Pi Data Archive ke arsitektur data modern. AWS Ini terintegrasi ke OSI Pi Asset Framework yang dikelola secara terpusat melalui pesan. AWS IoT PubSub

- Penyedia Parsec

  Komponen ini memungkinkan AWS IoT Greengrass perangkat untuk mengintegrasikan solusi keamanan perangkat keras menggunakan proyek Parsec open source dari Cloud Native Computing Foundation (CNCF).

- PostgreSQL DB

  Komponen ini menyediakan dukungan untuk database relasional PostgreSQL di tepi. Pelanggan dapat menggunakan komponen ini untuk menyediakan dan mengelola instance PostgreSQL lokal di dalam wadah docker.

- Pengunggah file S3

  Komponen ini memonitor direktori untuk file baru, mengunggahnya ke Amazon Simple Storage Service (Amazon S3), dan kemudian menghapusnya setelah unggahan berhasil.

- Secrets Manager klien

  Komponen ini menyediakan alat CLI yang dapat digunakan oleh komponen lain yang perlu mengambil rahasia dari komponen Secrets Manager dalam skrip siklus hidup resep.

- Perutean TES ke kontainer

  Komponen ini mengkonfigurasi nftables atau iptables pada AWS IoT Greengrass perangkat sehingga dapat menggunakan komponen dengan wadah. Layanan pertukaran token

- WebRTC

  Komponen ini menyerap aliran audio dan video dari kamera RTSP yang terhubung ke perangkat inti. AWS IoT Greengrass Dan kemudian komponen mengubah aliran audio dan video menjadi peer-to-peer komunikasi atau relay melalui Amazon Kinesis Video Streams.

Untuk meminta fitur atau melaporkan bug, buka GitHub masalah di repositori komponen tersebut. AWS tidak memberikan dukungan untuk komponen komunitas. Untuk informasi selengkapnya, lihat CONTRIBUTING.mdfile di repositori masing-masing komponen.

Beberapa komponen AWS yang disediakan juga open source. Untuk informasi selengkapnya, lihat Perangkat lunak inti AWS IoT Greengrass sumber terbuka.

AWS IoT Greengrassalat pengembangan

Gunakan alat AWS IoT Greengrass pengembangan untuk membuat, menguji, membangun, menerbitkan, dan menyebarkan komponen Greengrass kustom.

- Kit Pengembangan Greengrass CLI

  Gunakan AWS IoT Greengrass Development Kit Command-Line Interface (GDK CLI) di lingkungan pengembangan lokal Anda untuk membuat komponen dari template dan komponen komunitas di Katalog Perangkat Lunak Greengrass. Anda dapat menggunakan CLI GDK untuk membangun komponen dan mempublikasikan komponen ke layanan sebagai komponen pribadi AWS IoT Greengrass di Anda. Akun AWS

- Antarmuka Baris Perintah Greengrass

  Gunakan Antarmuka Baris Perintah Greengrass (Greengrass CLI) pada perangkat inti Greengrass untuk menyebarkan dan men-debug komponen Greengrass. CLI Greengrass adalah komponen yang dapat Anda terapkan ke perangkat inti Anda untuk membuat penerapan lokal, melihat detail tentang komponen yang diinstal, dan menjelajahi file log.

- Konsol debug lokal

  Gunakan konsol debug lokal di perangkat inti Greengrass untuk menyebarkan dan men-debug komponen Greengrass menggunakan antarmuka web dasbor lokal. Konsol debug lokal adalah komponen yang dapat Anda terapkan ke perangkat inti Anda untuk membuat penerapan lokal dan melihat detail tentang komponen yang diinstal.

AWS IoT Greengrassjuga menyediakan SDK berikut yang dapat Anda gunakan dalam komponen Greengrass khusus:

- PerpustakaanAWS IoT Device SDK, yang berisi perpustakaan komunikasi antarproses (IPC). Untuk informasi selengkapnya, lihat Gunakan AWS IoT Device SDK untuk berkomunikasi dengan inti Greengrass, komponen lain, dan AWS IoT Core.
- Stream Manager SDK, yang dapat Anda gunakan untuk mentransfer aliran data ke file. AWS Cloud Lihat informasi yang lebih lengkap di Kelola aliran data di perangkat inti Greengrass.

Topik

- AWS IoT GreengrassKit Pengembangan Antarmuka Baris Perintah
- Antarmuka Baris Perintah Greengrass
- Gunakan Kerangka AWS IoT Greengrass Pengujian

AWS IoT GreengrassKit Pengembangan Antarmuka Baris Perintah

AWS IoT GreengrassDevelopment Kit Command-Line Interface (GDK CLI) menyediakan fitur yang membantu Anda mengembangkan komponen Greengrass khusus. Anda dapat menggunakan CLI GDK untuk membuat, membangun, dan menerbitkan komponen kustom. Saat Anda membuat repositori komponen dengan CLI GDK, Anda dapat memulai dari template atau komponen komunitas dari Katalog Perangkat Lunak Greengrass. Kemudian, Anda dapat memilih sistem build yang mengemas file sebagai arsip ZIP, menggunakan skrip build Maven atau Gradle, atau menjalankan perintah build kustom. Setelah membuat komponen, Anda dapat menggunakan CLI GDK untuk mempublikasikannya ke AWS IoT Greengrass layanan, sehingga Anda dapat menggunakan AWS IoT Greengrass konsol atau API untuk menyebarkan komponen ke perangkat inti Greengrass Anda.

Saat Anda mengembangkan komponen Greengrass tanpa CLI GDK, Anda harus memperbarui versi dan URI artefak dalam file resep komponen setiap kali Anda membuat versi baru komponen. Saat Anda menggunakan CLI GDK, CLI dapat secara otomatis memperbarui versi dan URI artefak untuk Anda setiap kali Anda menerbitkan versi baru komponen.

CLI GDK adalah open source dan tersedia di. GitHub Anda dapat menyesuaikan dan memperluas CLI GDK untuk memenuhi kebutuhan pengembangan komponen Anda. Kami mengundang Anda untuk membuka masalah dan menarik permintaan pada GitHub repositori. Anda dapat menemukan sumber CLI GDK di tautan berikut: https://github.com/aws-greengrass/. aws-greengrass-gdk-cli

Prasyarat

Untuk menginstal dan menggunakan CLI Greengrass Development Kit, Anda memerlukan yang berikut ini:

- Sesi Akun AWS. Jika Anda tidak memilikinya, lihat Mengatur sebuah Akun AWS.
- Komputer pengembangan seperti Windows, macOS, atau Unix dengan koneksi internet.
- Untuk GDK CLI versi 1.1.0 atau yang lebih baru, Python 3.6 atau yang lebih baru diinstal pada komputer pengembangan Anda.

  Untuk GDK CLI versi 1.0.0, Python 3.8 atau yang lebih baru diinstal pada komputer pengembangan Anda.
- Git diinstal pada komputer pengembangan Anda.
- AWS Command Line Interface(AWS CLI) diinstal dan dikonfigurasi dengan kredensional di komputer pengembangan Anda. Untuk informasi selengkapnya, lihat Menginstal, memperbarui, dan melepas pemasangan AWS CLI dan Mengonfigurasi AWS CLI di Panduan Pengguna AWS Command Line Interface.

  > **Note**
  >
  > Jika Anda menggunakan Raspberry Pi atau perangkat ARM 32-bit lainnya, instal V1 AWS CLI. AWS CLI V2 tidak tersedia untuk perangkat ARM 32-bit. Untuk informasi selengkapnya, lihat Menginstal, memperbarui, dan mencopot instalasi AWS CLI versi 1.
- Untuk menggunakan CLI GDK untuk mempublikasikan komponen ke AWS IoT Greengrass layanan, Anda harus memiliki izin berikut:
  - `s3:CreateBucket`
  - `s3:GetBucketLocation`
  - `s3:PutObject`
  - `greengrass:CreateComponentVersion`
  - `greengrass:ListComponentVersions`
- Untuk menggunakan CLI GDK untuk membangun komponen yang artefaknya ada di bucket S3 dan bukan di sistem file lokal, Anda harus memiliki izin berikut:
  - `s3:ListBucket`

  Fitur ini tersedia untuk GDK CLI v1.1.0 dan yang lebih baru.

Changelog

Tabel berikut menjelaskan perubahan di setiap versi CLI GDK. Untuk informasi selengkapnya, lihat halaman Rilis CLI GDK di. GitHub

Versi	Perubahan
1.6.2	Perbaikan bug dan peningkatan • Memperbaiki masalah di mana Windows gradlew.bat tidak berfungsi karena jalur relatif. • Perbaikan kecil pada logging, pengujian, dan pengemasan.
1.6.1	Perbaikan bug dan peningkatan • Menambahkan perbaikan keamanan untuk penguraian argumen CLI. • Mengaktifkan GDK untuk mendapatkan nama rilis Greengrass Testing Framework (GTF) terbaru sebagai versi GTF default. • Memungkinkan GDK untuk merekomendasikan pelanggan menggunak an versi GTF yang lebih lama yang mereka perbarui ke versi terbaru.
1.6.0	Fitur baru • Menambahkan pemeriksaan validasi resep terhadap skema resep Greengrass selama perintah and. `component build` `component publish` Pembaruan ini membantu pengembang mengidentifikasi masalah yang dapat ditindaklanjuti dalam resep komponen mereka sebelumnya dalam proses pembuatan komponen. • Menambahkan suite uji kepercayaan ke template yang dapat ditarik ke bawah oleh `test-e2e init` perintah. Rangkaian uji kepercayaan diri ini mencakup delapan tes generik yang dapat digunakan dan diperluas agar sesuai dengan kebutuhan pengujian komponen dasar. Perbaikan bug dan peningkatan • Memperbarui versi Greengrass Testing Framework (GTF) default yang digunakan oleh perintah ke versi 1.2.0. `test-e2e`

Versi	Perubahan
1.5.0	Perbaikan bug dan peningkatan Memperbarui pola yang dikenali oleh opsi `excludes` build kapan `build_system zip`. Versi ini sekarang akan mengenali pola glob yang cocok dengan nama jalur berdasarkan karakter wildcard mereka. Ini memungkinkan spesifikasi khusus direktori mana yang akan dikecualikan.
1.4.0	Fitur baru • Menambahkan `config` perintah baru yang memulai prompt interaktif untuk memodifikasi bidang dalam file konfigurasi GDK yang ada. • Memodifikasi `gdk component publish` perintah `gdk component build` dan untuk memverifikasi bahwa ukuran resep berada dalam persyaratan Greengrass (<=16000 byte) sebelum melanjutkan. Perbaikan bug dan peningkatan • Menambahkan logging tambahan dalam output `gdk component build` perintah saat kesalahan sintaks resep mencegah build selesai untuk kesadaran. • Mengganti nama `otf-options` dan `otf-version` ke `gtf-options` dan `gtf-version` masing-masing, karena penggantian nama Open Test Framework menjadi Greengrass Testing Framework.
1.3.0	Fitur baru • Menambahkan `test-e2e` perintah baru untuk mendukung end-to-end pengujian komponen menggunakan Open Test Framework. • Menambahkan opsi konfigurasi baru,`zip_name`, untuk mendukung nama file zip yang dapat dikonfigurasi dengan sistem pembuatan zip. • Membuat `region` properti dalam file konfigurasi GDK opsional. Perbaikan bug dan peningkatan • Memperbaiki masalah di mana direktori baru dibuat bahkan ketika template atau repositori yang ditentukan tidak ada saat menginisialisasi proyek GDK dengan argumen. `--name`

Versi	Perubahan
1.2.3	Perbaikan bug dan peningkatan • Memperbaiki masalah saat pembuatan bucket gagal karena penangana n kesalahan yang salah. • Memperbaiki masalah di mana struktur daftar dalam resep komponen dihapus.
1.2.2	Perbaikan bug dan peningkatan • Kunci resep tidak lagi peka huruf besar/kecil. • Menambahkan pemeriksaan untuk menentukan apakah ada bucket di Wilayah AWS dan dapat diakses oleh pengguna sebelum membuat bucket baru. Membutuhkan pengguna untuk memiliki `GetBucket Location` izin. • Memperbaiki masalah dengan `excludes` kata kunci dalam file konfigura si CLI GDK.
1.2.1	Perbaikan bug dan peningkatan • Menerima Kanada (Tengah) (`ca-central-1`) Wilayah AWS di entri konfigurasi wilayah dalam `gdk-config.json` file. • Memperbaiki masalah dengan argumen `--region` CLI GDK ke perintah. `publish`

Versi	Perubahan
1.2.0	Fitur baru • Menambahkan `options` entri ke `build` konfigurasi dalam file konfigurasi CLI GDK. Mendukung `excludes` under `options` untuk mengecualikan file tertentu dari artefak zip saat menggunakan sistem `zip` build. • Menambahkan sistem `gradlew` build untuk menggunakan Gradle Wrapper untuk membangun komponen. • Menambahkan dukungan untuk file build DSL Kotlin untuk opsi `gradle` build. • Menambahkan `options` entri ke `publish` konfigurasi dalam file konfigurasi CLI GDK. Mendukung `file_upload_args` under `options` untuk memberikan argumen tambahan saat mengunggah file ke Amazon S3. Perbaikan bug dan peningkatan • Memperbaiki masalah saat build Gradle tidak dibersihkan sebelum menjalankan perintah build. • Memperbaiki masalah saat build tidak keluar saat perintah build gagal. • Meningkatkan format output `gdk component list` perintah.

Versi	Perubahan
1.1.0	Fitur baru • Menambahkan dukungan untuk sistem build Gradle. • Menambahkan dukungan untuk sistem build Maven di perangkat Windows. • Menambahkan `--bucket` argumen ke perintah component publish. Anda dapat menggunakan argumen ini untuk menentukan bucket yang tepat di mana CLI GDK mengunggah artefak komponen. • Menambahkan `--name` argumen ke perintah komponen init. Anda dapat menggunakan opsi ini untuk menentukan folder tempat CLI GDK menginisialisasi komponen. • Menambahkan dukungan untuk artefak komponen yang ada di bucket S3 tetapi tidak di folder build komponen lokal. Anda dapat menggunak an fitur ini untuk mengurangi biaya bandwidth untuk artefak komponen besar, seperti model pembelajaran mesin. Perbaikan bug dan peningkatan • Memperbarui perintah component publish untuk memeriksa apakah komponen dibangun sebelum menerbitkan komponen. Jika komponen tidak dibangun, perintah ini sekarang membangun komponen untuk Anda. • Memperbaiki masalah di mana sistem pembuatan zip gagal dibangun di perangkat Windows saat nama file ZIP berisi huruf kapital. • Meningkatkan format pesan log dan mengubah tingkat log default ke `INFO` perangkat yang menjalankan versi Python lebih awal dari 3.8. • Mengubah persyaratan versi Python minimum ke Python 3.6.
1.0.0	Versi awal.

Instal atau perbarui Antarmuka AWS IoT Greengrass Baris Perintah Kit Pengembangan

AWS IoT GreengrassDevelopment Kit Command-Line Interface (GDK CLI) dibangun di atas Python, sehingga Anda dapat `pip` menggunakannya untuk menginstalnya di komputer pengembangan Anda.

> **Tip**
>
> Anda juga dapat menginstal CLI GDK di lingkungan virtual Python seperti venv. Untuk informasi selengkapnya, lihat Lingkungan dan Paket Virtual dalam dokumentasi Python 3.

Untuk menginstal atau memperbarui CLI GDK

1. Jalankan perintah berikut untuk menginstal versi terbaru CLI GDK dari GitHub repositorinya.

   ```
   python3 -m pip install -U git+https://github.com/aws-greengrass/aws-greengrass-gdk-cli.git@v1.6.2
   ```

   > **Note**
   >
   > Untuk menginstal versi tertentu dari CLI GDK, *ganti* VersionTag dengan tag versi yang akan diinstal. Anda dapat melihat tag versi untuk CLI GDK di GitHub repositorinya.
   >
   > ```
   > python3 -m pip install -U git+https://github.com/aws-greengrass/aws-greengrass-gdk-cli.git@versionTag
   > ```

2. Jalankan perintah berikut untuk memverifikasi bahwa CLI GDK berhasil diinstal.

   ```
   gdk --help
   ```

   Jika `gdk` perintah tidak ditemukan, tambahkan foldernya ke PATH.

   - Pada perangkat Linux, tambahkan `/home/`*`MyUser`*`/.local/bin` ke PATH, dan ganti *`MyUser`*dengan nama pengguna Anda.
   - Pada perangkat Windows, tambahkan *`PythonPath`*`\\Scripts` ke PATH, dan ganti *`PythonPath`*dengan jalur ke folder Python di perangkat Anda.

Anda sekarang dapat menggunakan CLI GDK untuk membuat, membangun, dan menerbitkan komponen Greengrass. Untuk informasi selengkapnya tentang cara menggunakan CLI GDK, lihat. AWS IoT GreengrassPerintah Antarmuka Baris Perintah Kit Pengembangan

AWS IoT GreengrassPerintah Antarmuka Baris Perintah Kit Pengembangan

AWS IoT GreengrassDevelopment Kit Command-Line Interface (GDK CLI) menyediakan antarmuka baris perintah yang dapat Anda gunakan untuk membuat, membangun, dan menerbitkan komponen Greengrass di komputer pengembangan Anda. Perintah CLI GDK menggunakan format berikut.

```
gdk <command> <subcommand> [arguments]
```

Saat Anda menginstal CLI GDK, penginstal `gdk` menambahkan ke PATH sehingga Anda dapat menjalankan CLI GDK dari baris perintah.

Anda dapat menggunakan argumen berikut dengan perintah apa pun:

- Gunakan `-h` atau `--help` untuk informasi tentang perintah CLI GDK.
- Gunakan `-v` atau `--version` untuk melihat versi CLI GDK apa yang diinstal.
- Gunakan `-d` atau `--debug` untuk menampilkan log verbose yang dapat Anda gunakan untuk men-debug CLI GDK.

Bagian ini menjelaskan perintah CLI GDK dan memberikan contoh untuk setiap perintah. Sinopsis untuk setiap perintah menunjukkan argumen dan penggunaannya. Argumen opsional ditampilkan dalam tanda kurung siku.

Perintah yang tersedia

- komponen
- config
- tes-e2e

komponen

Gunakan `component` perintah di AWS IoT Greengrass Development Kit Command-Line Interface (GDK CLI) untuk membuat, membangun, dan menerbitkan komponen Greengrass kustom.

Subperintah

- inisialisasi
- build
- publish

- daftar

inisialisasi

Menginisialisasi folder komponen Greengrass dari template komponen atau komponen komunitas.

CLI GDK mengambil komponen komunitas dari Katalog Perangkat Lunak Greengrass dan templat komponen dari repositori Template Komponen pada. AWS IoT Greengrass GitHub

> **Note**
>
> Jika Anda menggunakan GDK CLI v1.0.0, Anda harus menjalankan perintah ini di folder kosong. CLI GDK mengunduh template atau komponen komunitas ke folder saat ini.
> Jika Anda menggunakan GDK CLI v1.1.0 atau yang lebih baru, Anda dapat menentukan `--name` argumen untuk menentukan folder tempat CLI GDK mengunduh template atau komponen komunitas. Jika Anda menggunakan argumen ini, tentukan folder yang tidak ada. CLI GDK membuat folder untuk Anda. Jika Anda tidak menentukan argumen ini, CLI GDK menggunakan folder saat ini, yang harus kosong.
> Jika komponen menggunakan sistem pembuatan zip, CLI GDK membuat zip file tertentu di folder komponen ke dalam file zip dengan nama yang sama dengan folder komponen. Misalnya, jika nama folder komponen adalah`HelloWorld`, CLI GDK membuat file zip bernama. `HelloWorld.zip` Dalam resep komponen, nama artefak zip harus cocok dengan nama folder komponen. Jika Anda menggunakan GDK CLI versi 1.0.0 pada perangkat Windows, folder komponen dan nama file zip harus berisi hanya huruf kecil.
> Jika Anda menginisialisasi template atau komponen komunitas yang menggunakan sistem pembuatan zip ke folder dengan nama yang berbeda dari template atau komponen, Anda harus mengubah nama artefak zip dalam resep komponen. Perbarui `Artifacts` dan `Lifecycle` definisi sedemikian rupa sehingga nama file zip cocok dengan nama folder komponen. Contoh berikut menyoroti nama file zip dalam `Artifacts` dan `Lifecycle` definisi.
>
> JSON
>
> ```
> {
>   ...
>   "Manifests": [
>     {
>       "Platform": {
>         "os": "all"
> ```

```
      },
      "Artifacts": [
        {
          "URI": "s3://BUCKET_NAME/COMPONENT_NAME/
COMPONENT_VERSION/HelloWorld.zip",
          "Unarchive": "ZIP"
        }
      ],
      "Lifecycle": {
        "run": "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
 {configuration:/Message}"
      }
    }
  ]
}
```

YAML

```
---
...
Manifests:
  - Platform:
      os: all
    Artifacts:
      - URI: "s3://BUCKET_NAME/COMPONENT_NAME/
COMPONENT_VERSION/HelloWorld.zip"
        Unarchive: ZIP
    Lifecycle:
      run: "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
 {configuration:/Message}"
```

Sinopsis

```
$ gdk component init
    [--language]
    [--template]
    [--repository]
    [--name]
```

Argumen (inisialisasi dari template komponen)

- `-l`, `--language` — Bahasa pemrograman yang digunakan untuk template yang Anda tentukan.

  Anda harus menentukan salah satu `--repository` atau `--language` dan`--template`.

- `-t`, `--template` — Template komponen yang digunakan untuk proyek komponen lokal. Untuk melihat template yang tersedia, gunakan perintah list.

  Anda harus menentukan salah satu `--repository` atau `--language` dan`--template`.

- `-n`, `--name` — (Opsional) Nama folder lokal tempat CLI GDK menginisialisasi komponen. Tentukan folder yang tidak ada. CLI GDK membuat folder untuk Anda.

  Fitur ini tersedia untuk GDK CLI v1.1.0 dan yang lebih baru.

Argumen (inisialisasi dari komponen komunitas)

- `-r`, `--repository` — Komponen komunitas untuk memeriksa ke folder lokal. Untuk melihat komponen komunitas yang tersedia, gunakan perintah list.

  Anda harus menentukan salah satu `--repository` atau `--language` dan`--template`.

- `-n`, `--name` — (Opsional) Nama folder lokal tempat CLI GDK menginisialisasi komponen. Tentukan folder yang tidak ada. CLI GDK membuat folder untuk Anda.

  Fitur ini tersedia untuk GDK CLI v1.1.0 dan yang lebih baru.

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini untuk menginisialisasi folder komponen dari template Python Hello World.

```
$ gdk component init -l python -t HelloWorld
[2021-11-29 12:51:40] INFO - Initializing the project directory with a python
 component template - 'HelloWorld'.
[2021-11-29 12:51:40] INFO - Fetching the component template 'HelloWorld-python'
 from Greengrass Software Catalog.
```

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini untuk menginisialisasi folder komponen dari komponen komunitas.

```
$ gdk component init -r aws-greengrass-labs-database-influxdb
```

```
[2022-01-24 15:44:33] INFO - Initializing the project directory with a component
 from repository catalog - 'aws-greengrass-labs-database-influxdb'.
[2022-01-24 15:44:33] INFO - Fetching the component repository 'aws-greengrass-labs-
database-influxdb' from Greengrass Software Catalog.
```

build

Bangun sumber komponen menjadi resep dan artefak yang dapat Anda publikasikan ke AWS IoT Greengrass layanan. CLI GDK menjalankan sistem build yang Anda tentukan dalam file konfigurasi CLI GDK,. `gdk-config.json` Anda harus menjalankan perintah ini di folder yang sama di mana `gdk-config.json` file itu ada.

Saat Anda menjalankan perintah ini, CLI GDK membuat resep dan artefak di folder `greengrass-build` di folder komponen. CLI GDK menyimpan resep di `greengrass-build/recipes` folder dan menyimpan artefak di folder. `greengrass-build/artifacts/componentName/componentVersion`

Jika Anda menggunakan GDK CLI v1.1.0 atau yang lebih baru, resep komponen dapat menentukan artefak yang ada di bucket S3 tetapi tidak di folder build komponen lokal. Anda dapat menggunakan fitur ini untuk mengurangi penggunaan bandwidth ketika Anda mengembangkan komponen dengan artefak besar, seperti model pembelajaran mesin.

Setelah membuat komponen, Anda dapat melakukan salah satu hal berikut untuk mengujinya di perangkat inti Greengrass:

- Jika Anda mengembangkan pada perangkat yang berbeda dari tempat Anda menjalankan perangkat lunak AWS IoT Greengrass Core, Anda harus mempublikasikan komponen untuk menerapkannya ke perangkat inti Greengrass. Publikasikan komponen ke AWS IoT Greengrass layanan, dan terapkan ke perangkat inti Greengrass. Untuk informasi selengkapnya, lihat perintah publish danBuat deployment.
- Jika Anda mengembangkan pada perangkat yang sama di mana Anda menjalankan perangkat lunak AWS IoT Greengrass Core, Anda dapat mempublikasikan komponen ke AWS IoT Greengrass layanan yang akan digunakan, atau Anda dapat membuat penyebaran lokal untuk menginstal dan menjalankan komponen. Untuk membuat penerapan lokal, gunakan CLI Greengrass. Lihat informasi yang lebih lengkap di Antarmuka Baris Perintah Greengrass dan Uji AWS IoT Greengrass komponen dengan penerapan lokal. Saat Anda membuat penyebaran lokal, tentukan `greengrass-build/recipes` sebagai folder resep dan `greengrass-build/artifacts` sebagai folder artefak.

Sinopsis

```
$ gdk component build
```

Argumen

Tidak ada

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ gdk component build
[2021-11-29 13:18:49] INFO - Getting project configuration from gdk-config.json
[2021-11-29 13:18:49] INFO - Found component recipe file 'recipe.yaml' in the
 project directory.
[2021-11-29 13:18:49] INFO - Building the component 'com.example.PythonHelloWorld'
 with the given project configuration.
[2021-11-29 13:18:49] INFO - Using 'zip' build system to build the component.
[2021-11-29 13:18:49] WARNING - This component is identified as using 'zip' build
 system. If this is incorrect, please exit and specify custom build command in the
 'gdk-config.json'.
[2021-11-29 13:18:49] INFO - Zipping source code files of the component.
[2021-11-29 13:18:49] INFO - Copying over the build artifacts to the greengrass
 component artifacts build folder.
[2021-11-29 13:18:49] INFO - Updating artifact URIs in the recipe.
[2021-11-29 13:18:49] INFO - Creating component recipe in 'C:\Users\MyUser\Documents
\greengrass-components\python\HelloWorld\greengrass-build\recipes'.
```

publish

Publikasikan komponen ini ke AWS IoT Greengrass layanan. Perintah ini mengunggah artefak build ke bucket S3, memperbarui URI artefak dalam resep, dan membuat versi komponen baru dari resep. CLI GDK menggunakan bucket S3 AWS dan Wilayah yang Anda tentukan dalam file konfigurasi CLI GDK,. `gdk-config.json` Anda harus menjalankan perintah ini di folder yang sama di mana `gdk-config.json` file itu ada.

Jika Anda menggunakan GDK CLI v1.1.0 atau yang lebih baru, Anda dapat menentukan argumen untuk menentukan `--bucket` bucket S3 tempat CLI GDK mengunggah artefak komponen. Jika Anda tidak menentukan argumen ini, CLI GDK akan mengunggah ke bucket S3 yang namanya, di *mana* bucket *dan region bucket-region-accountId adalah nilai yang Anda*

tentukangdk-config.json, dan accountID adalah ID Anda. Akun AWS CLI GDK membuat bucket jika tidak ada.

Jika Anda menggunakan GDK CLI v1.2.0 atau yang lebih baru, Anda dapat mengganti yang ditentukan dalam file Wilayah AWS konfigurasi CLI GDK menggunakan parameter. `--region` Anda juga dapat menentukan opsi tambahan menggunakan `--options` parameter. Untuk daftar opsi yang tersedia, lihatFile konfigurasi CLI Kit Pengembangan Greengrass.

Saat Anda menjalankan perintah ini, CLI GDK menerbitkan komponen dengan versi yang Anda tentukan dalam resep. Jika Anda menentukan`NEXT_PATCH`, CLI GDK menggunakan versi patch berikutnya yang belum ada. Versi semantik menggunakan mayor. kecil. sistem penomoran patch. Untuk informasi lebih lanjut, lihat spesifikasi versi semantik.

> **Note**
>
> Jika Anda menggunakan GDK CLI v1.1.0 atau yang lebih baru, saat Anda menjalankan perintah ini, CLI GDK akan memeriksa apakah komponen tersebut dibangun. Jika komponen tidak dibangun, CLI GDK akan membangun komponen sebelum menerbitkan komponen.

Sinopsis

```
$ gdk component publish
    [--bucket] [--region] [--options]
```

Argumen

- `-b`, `--bucket` — (Opsional) Tentukan nama bucket S3 tempat CLI GDK menerbitkan artefak komponen.

  Jika Anda tidak menentukan argumen ini, CLI GDK akan mengunggah ke bucket S3 yang namanya, di *mana* bucket *dan region bucket-region-accountId adalah nilai yang Anda tentukangdk-config.json, dan accountID* adalah ID Anda. Akun AWS CLI GDK membuat bucket jika tidak ada.

  CLI GDK membuat bucket jika tidak ada.

  Fitur ini tersedia untuk GDK CLI v1.1.0 dan yang lebih baru.
- `-r`, `--region` — (Opsional) Tentukan nama Wilayah AWS to saat komponen dibuat. Argumen ini mengesampingkan nama Wilayah dalam konfigurasi CLI GDK.

Fitur ini tersedia untuk GDK CLI v1.2.0 dan yang lebih baru.

- `-o`, `--options` (Opsional) Tentukan daftar opsi untuk menerbitkan komponen. Argumen harus berupa string JSON yang valid atau path file ke file JSON yang berisi opsi penerbitan. Argumen ini mengesampingkan opsi dalam konfigurasi CLI GDK.

  Fitur ini tersedia untuk GDK CLI v1.2.0 dan yang lebih baru.

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ gdk component publish
[2021-11-29 13:45:29] INFO - Getting project configuration from gdk-config.json
[2021-11-29 13:45:29] INFO - Found component recipe file 'recipe.yaml' in the
 project directory.
[2021-11-29 13:45:29] INFO - Found credentials in shared credentials file: ~/.aws/
credentials
[2021-11-29 13:45:30] INFO - Publishing the component 'com.example.PythonHelloWorld'
 with the given project configuration.
[2021-11-29 13:45:30] INFO - No private version of the component
 'com.example.PythonHelloWorld' exist in the account. Using '1.0.0' as the next
 version to create.
[2021-11-29 13:45:30] INFO - Uploading the component built artifacts to s3 bucket.
[2021-11-29 13:45:30] INFO - Uploading component artifacts to S3 bucket: {bucket}.
 If this is your first time using this bucket, add the 's3:GetObject' permission
 to each core device's token exchange role to allow it to download the component
 artifacts. For more information, see https://docs.aws.amazon.com/greengrass/v2/
developerguide/device-service-role.html.
[2021-11-29 13:45:30] INFO - Not creating an artifacts bucket as it already exists.
[2021-11-29 13:45:30] INFO - Updating the component recipe
 com.example.PythonHelloWorld-1.0.0.
[2021-11-29 13:45:30] INFO - Creating a new greengrass component
 com.example.PythonHelloWorld-1.0.0
[2021-11-29 13:45:30] INFO - Created private version '1.0.0' of the component in the
 account.'com.example.PythonHelloWorld'.
```

daftar

Ambil daftar template komponen dan komponen komunitas yang tersedia.

CLI GDK mengambil komponen komunitas dari Katalog Perangkat Lunak Greengrass dan templat komponen dari repositori Template Komponen pada. AWS IoT Greengrass GitHub

Anda dapat meneruskan output dari perintah ini ke perintah init untuk menginisialisasi repositori komponen dari template dan komponen komunitas.

Sinopsis

```
$ gdk component list
    [--template]
    [--repository]
```

Argumen

- `-t`, `--template` — (Opsional) Tentukan argumen ini untuk mencantumkan templat komponen yang tersedia. Perintah ini menampilkan nama dan bahasa dari setiap template dalam format*`name`*-*`language`*. Misalnya, di`HelloWorld-python`, nama template adalah `HelloWorld` dan bahasanya`python`.
- `-r`, `--repository` — (Opsional) Tentukan argumen ini untuk daftar repositori komponen komunitas yang tersedia.

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ gdk component list --template
[2021-11-29 12:29:04] INFO - Listing all the available component templates from
 Greengrass Software Catalog.
[2021-11-29 12:29:04] INFO - Found '2' component templates to display.
1. HelloWorld-python
2. HelloWorld-java
```

config

Gunakan `config` perintah di AWS IoT Greengrass Development Kit Command-Line Interface (GDK CLI) untuk memodifikasi konfigurasi untuk GDK dalam file konfigurasi,. `gdk-config.json`

Subperintah

- update

update

Mulai prompt interaktif untuk memodifikasi bidang dalam file konfigurasi GDK yang ada.

Sinopsis

```
$ gdk config update
    [--component]
```

Argumen

- `-c`, `--component` — Untuk memperbarui bidang terkait komponen dalam file. `gdk-config.json` Argumen ini diperlukan karena ini adalah satu-satunya pilihan.

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini untuk mengkonfigurasi komponen.

```
$ gdk config update --component
Current value of the REQUIRED component_name is (default:
 com.example.PythonHelloWorld):
Current value of the REQUIRED author is (default: author):
Current value of the REQUIRED version is (default: NEXT_PATCH):
Do you want to change the build configurations? (y/n)
Do you want to change the publish configurations? (y/n)
[2023-09-26 10:19:48] INFO - Config file has been updated. Exiting...
```

tes-e2e

Gunakan `test-e2e` perintah dalam AWS IoT Greengrass Development Kit Command-Line Interface (GDK CLI) untuk menginisialisasi, membangun, dan end-to-end menjalankan modul pengujian dalam proyek GDK.

Subperintah

- inisialisasi
- build
- run

inisialisasi

Inisialisasi proyek CLI GDK yang ada dengan modul pengujian yang menggunakan Greengrass Testing Framework (GTF).

Secara default, CLI GDK mengambil template modul maven dari AWS IoT Greengrass repositori Component Templates pada. GitHub Modul maven ini dilengkapi dengan ketergantungan pada file JAR. `aws-greengrass-testing-standalone`

Perintah ini membuat direktori baru yang disebut `gg-e2e-tests` di dalam proyek GDK. Jika direktori modul pengujian sudah ada dan tidak kosong, perintah keluar tanpa melakukan apa pun. `gg-e2e-tests`Folder ini berisi fitur Mentimun dan definisi langkah yang terstruktur dalam proyek maven.

Secara default, perintah ini akan mencoba menggunakan versi rilis terbaru dari GTF.

Sinopsis

```
$ gdk test-e2e init
    [--gtf-version]
```

Argumen

- `-ov`, `--gtf-version` — (Opsional) Versi GTF untuk digunakan dengan modul end-to-end pengujian dalam proyek GDK. Nilai ini harus menjadi salah satu versi GTF dari rilis. Argumen ini mengesampingkan konfigurasi CLI GDK. `gtf_version`

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini untuk menginisialisasi proyek GDK dengan modul pengujian.

```
$ gdk test-e2e init
[2023-12-06 12:20:28] INFO - Using the GTF version provided in the GDK test config
 1.2.0
[2023-12-06 12:20:28] INFO - Downloading the E2E testing template from GitHub into
 gg-e2e-tests directory...
```

build

> **Note**
>
> Anda harus membangun komponen dengan menjalankan gdk component build sebelum membangun modul end-to-end pengujian.

Bangun modul end-to-end pengujian. CLI GDK membangun modul pengujian menggunakan sistem build yang Anda tentukan dalam file konfigurasi CLI GDK, di bawah properti. `gdk-config.json` `test-e2e` Anda harus menjalankan perintah ini di folder yang sama di mana `gdk-config.json` file itu ada.

Secara default, CLI GDK menggunakan sistem build maven untuk membangun modul pengujian. Maven diperlukan untuk menjalankan perintah. `gdk test-e2e build`

Anda harus membangun komponen dengan menjalankan gdk-component-build sebelum membangun modul pengujian, jika file fitur pengujian memiliki variabel seperti `GDK_COMPONENT_NAME` dan `GDK_COMPONENT_RECIPE_FILE` untuk diinterpolasi.

Saat Anda menjalankan perintah ini, CLI GDK menginterpolasi semua variabel dari konfigurasi proyek GDK dan membangun modul untuk menghasilkan file JAR `gg-e2e-tests` pengujian akhir.

Sinopsis

```
$ gdk test-e2e build
```

Argumen

Tidak ada

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ gdk test-e2e build
[2023-07-20 15:36:48] INFO - Updating feature file: file:///path/to//
HelloWorld/greengrass-build/gg-e2e-tests/src/main/resources/greengrass/features/
component.feature
[2023-07-20 15:36:48] INFO - Creating the E2E testing recipe file:///path/to/
HelloWorld/greengrass-build/recipes/e2e_test_recipe.yaml
[2023-07-20 15:36:48] INFO - Building the E2E testing module
```

```
[2023-07-20 15:36:48] INFO - Running the build command 'mvn package'
.........
```

run

Jalankan modul pengujian dengan opsi pengujian di file konfigurasi GDK.

> **Note**
>
> Anda harus membangun modul pengujian dengan menjalankan gdk test-e2e build sebelum menjalankan end-to-end tes.

Sinopsis

```
$ gdk test-e2e run
    [--gtf-options]
```

Argumen

- `-oo`, `--gtf-options` — (Opsional) Tentukan daftar opsi untuk menjalankan end-to-end tes. Argumen harus berupa string JSON yang valid atau path file ke file JSON yang berisi opsi GTF. Opsi yang disediakan dalam file konfigurasi digabungkan dengan yang disediakan dalam argumen perintah. Jika opsi hadir di kedua tempat, yang ada dalam argumen lebih diutamakan daripada yang dari file konfigurasi.

  Jika `tags` opsi tidak ditentukan dalam perintah ini, GDK menggunakan `Sample` tag. Jika tidak `ggc-archive` ditentukan, GDK mengunduh versi terbaru dari arsip inti Greengrass.

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ gdk test-e2e run
[2023-07-20 16:35:53] INFO - Downloading latest nucleus archive from url https://
d2s8p88vqu9w66.cloudfront.net/releases/greengrass-latest.zip
[2023-07-20 16:35:57] INFO - Running test jar with command java -jar /path/to/
greengrass-build/gg-e2e-tests/target/uat-features-1.0.0.jar –ggc-archive=/path/to/
aws-greengrass-gdk-cli/HelloWorld/greengrass-build/greengrass-nucleus-latest.zip –
tags=Sample
```

```
16:35:59.693 [] [] [] [INFO]
 com.aws.greengrass.testing.modules.GreengrassContextModule - Extracting /path/
to/workplace/aws-greengrass-gdk-cli/HelloWorld/greengrass-build/greengrass-
nucleus-latest.zip into /var/folders/7g/ltzcb_3s77nbtmkzfb6brwv40000gr/T/gg-
testing-7718418114158172636/greengrass
16:36:00.534 [gtf-1.1.0-SNAPSHOT] [] [] [INFO]
 com.aws.greengrass.testing.features.LoggerSteps - GTF Version is gtf-1.1.0-SNAPSHOT
.......
```

File konfigurasi CLI Kit Pengembangan Greengrass

AWS IoT GreengrassDevelopment Kit Command-Line Interface (GDK CLI) membaca dari file konfigurasi `gdk-config.json` bernama untuk membangun dan menerbitkan komponen. File konfigurasi ini harus ada di root repositori komponen. Anda dapat menggunakan perintah GDK CLI init untuk menginisialisasi repositori komponen dengan file konfigurasi ini.

Topik

- Format file konfigurasi GDK CLI
- Contoh file konfigurasi GDK CLI

Format file konfigurasi GDK CLI

Saat Anda menentukan file konfigurasi CLI GDK untuk komponen, Anda menentukan informasi berikut dalam format JSON.

`gdk_version`

Versi minimum CLI GDK yang kompatibel dengan komponen ini. Nilai ini harus menjadi salah satu versi CLI GDK dari rilis.

`component`

Konfigurasi untuk komponen ini.

`componentName`

`author`

Penulis atau penerbit komponen.

`version`

Versi komponen. Tentukan satu dari yang berikut ini:

- `NEXT_PATCH`— Saat Anda memilih opsi ini, CLI GDK menetapkan versi saat Anda mempublikasikan komponen. CLI GDK menanyakan layanan AWS IoT Greengrass untuk mengidentifikasi versi komponen terbaru yang diterbitkan. Kemudian, ia menetapkan versi ke versi patch berikutnya setelah versi itu. Jika Anda belum mempublikasikan komponen sebelumnya, CLI GDK menggunakan versi. `1.0.0`

  Jika Anda memilih opsi ini, Anda tidak dapat menggunakan CLI Greengrass untuk menyebarkan dan menguji komponen secara lokal ke komputer pengembangan lokal Anda yang menjalankan perangkat lunak Core. AWS IoT Greengrass Untuk mengaktifkan penerapan lokal, Anda harus menentukan versi semantik sebagai gantinya.

- Versi semantik, seperti. **`1.0.0`** Versi semantik menggunakan mayor. kecil. sistem penomoran patch. Untuk informasi lebih lanjut, lihat spesifikasi versi semantik.

  Jika Anda mengembangkan komponen pada perangkat inti Greengrass tempat Anda ingin menerapkan dan menguji komponen, pilih opsi ini. Anda harus membangun komponen dengan versi tertentu untuk membuat penerapan lokal dengan CLI Greengrass.

`build`

Konfigurasi yang digunakan untuk membangun sumber komponen ini menjadi artefak. Objek ini berisi informasi berikut:

`build_system`

Sistem build yang akan digunakan. Pilih dari salah satu pilihan berikut:

- `zip`— Mengemas folder komponen ke dalam file ZIP untuk didefinisikan sebagai satu-satunya artefak komponen. Pilih opsi ini untuk jenis komponen berikut:
  - Komponen yang menggunakan bahasa pemrograman yang ditafsirkan, seperti JavaScript Python atau.
  - Komponen yang mengemas file selain kode, seperti model pembelajaran mesin atau sumber daya lainnya.

  CLI GDK meritsleting folder komponen ke dalam file zip dengan nama yang sama dengan folder komponen. Misalnya, jika nama folder komponen adalah`HelloWorld`, CLI GDK membuat file zip bernama. `HelloWorld.zip`

> **Note**
>
> Jika Anda menggunakan GDK CLI versi 1.0.0 pada perangkat Windows, folder komponen dan nama file zip harus berisi hanya huruf kecil.

Ketika CLI GDK meritsleting folder komponen ke dalam file zip, ia melewatkan file-file berikut:

- File `gdk-config.json`
- File resep (`recipe.jsonataurecipe.yaml`)
- Membangun folder, seperti `greengrass-build`

- `maven`— Menjalankan `mvn clean package` perintah untuk membangun sumber komponen menjadi artefak. Pilih opsi ini untuk komponen yang menggunakan Maven, seperti komponen Java.

  Pada perangkat Windows, fitur ini tersedia untuk GDK CLI v1.1.0 dan yang lebih baru.

- `gradle`— Menjalankan `gradle build` perintah untuk membangun sumber komponen menjadi artefak. Pilih opsi ini untuk komponen yang menggunakan Gradle. Fitur ini tersedia untuk GDK CLI v1.1.0 dan yang lebih baru.

  Sistem `gradle` build mendukung Kotlin DSL sebagai file build. Fitur ini tersedia untuk GDK CLI v1.2.0 dan yang lebih baru.

- `gradlew`— Menjalankan `gradlew` perintah untuk membangun sumber komponen menjadi artefak. Pilih opsi ini untuk komponen yang menggunakan Gradle Wrapper.

  Fitur ini tersedia untuk GDK CLI v1.2.0 dan yang lebih baru.

- `custom`— Menjalankan perintah khusus untuk membangun sumber komponen menjadi resep dan artefak. Tentukan perintah khusus dalam `custom_build_command` parameter.

`custom_build_command`

(Opsional) Perintah custom build untuk dijalankan untuk sistem build kustom. Anda harus menentukan parameter ini jika Anda menentukan `custom` untuk`build_system`.

> **Important**
>
> Perintah ini harus membuat resep dan artefak di folder berikut dalam folder komponen. CLI GDK membuat folder ini untuk Anda saat Anda menjalankan perintah build [komponen].
>
> - Folder resep: `greengrass-build/recipes`
> - Folder artefak: `greengrass-build/artifacts/componentName/componentVersion`
>
>   Ganti *componentName* dengan nama komponen, dan ganti *componentVersion dengan versi* komponen atau. `NEXT_PATCH`

Anda dapat menentukan satu string atau daftar string, di mana setiap string adalah kata dalam perintah. Misalnya, untuk menjalankan perintah build kustom untuk komponen C++, Anda dapat menentukan **`cmake --build build --config Release`** atau**`["cmake", "--build", "build", "--config", "Release"]`**.

Untuk melihat contoh sistem build kustom, lihat [aws.greengrass.labs.LocalWebServer community componentdi GitHub].

`options`

(Opsional) Opsi konfigurasi tambahan yang digunakan selama proses pembuatan komponen.

Fitur ini tersedia untuk GDK CLI v1.2.0 dan yang lebih baru.

`excludes`

Daftar pola glob yang menentukan file mana yang akan dikecualikan dari direktori komponen saat membuat file zip. Hanya berlaku ketika `build_system` adazip.

> **Note**
>
> Di GDK CLI versi 1.4.0 dan sebelumnya, file apa pun yang cocok dengan entri dalam daftar pengecualian dikecualikan dari semua subdirektori komponen. Untuk mencapai perilaku yang sama di GDK CLI versi 1.5.0 dan yang lebih baru, tambahkan `**/` entri yang ada dalam daftar pengecualian.

Misalnya, `*.txt` akan mengecualikan file teks hanya dari direktori; `**/*.txt` akan mengecualikan file teks dari semua direktori dan subdirektori. Di GDK CLI versi 1.5.0 dan yang lebih baru, Anda mungkin melihat peringatan selama pembuatan komponen `excludes` saat ditentukan dalam file konfigurasi GDK. Untuk menonaktifkan peringatan ini, atur variabel lingkungan `GDK_EXCLUDES_WARN_IGNORE` `ketrue`.

CLI GDK selalu mengecualikan file berikut dari file zip:

- File `gdk-config.json`
- File resep (`recipe.jsonataurecipe.yaml`)
- Membangun folder, seperti `greengrass-build`

File-file berikut dikecualikan secara default. Namun, Anda dapat mengontrol file mana yang dikecualikan dengan `excludes` opsi.

- Setiap folder yang dimulai dengan awalan “test” () `test*`
- Semua file tersembunyi
- `node_modules`Folder

Jika Anda menentukan `excludes` opsi, CLI GDK hanya mengecualikan file-file yang Anda atur dengan opsi. `excludes` Jika Anda tidak menentukan `excludes` opsi, CLI GDK mengecualikan file dan folder default yang disebutkan sebelumnya.

`zip_name`

Nama file zip yang akan digunakan saat Anda membuat artefak zip selama proses pembuatan. Hanya berlaku ketika `build_system` adazip. Jika kosong, nama komponen digunakan untuk nama file zip. `build_system`

`publish`

Konfigurasi yang digunakan untuk mempublikasikan komponen ini ke AWS IoT Greengrass layanan.

Jika Anda menggunakan GDK CLI v1.1.0 atau yang lebih baru, Anda dapat menentukan argumen untuk menentukan `--bucket` bucket S3 tempat CLI GDK mengunggah artefak komponen. Jika Anda tidak menentukan argumen ini, CLI GDK akan mengunggah ke

bucket S3 yang namanya, di *mana* bucket *dan region bucket-region-accountId adalah nilai yang Anda tentukangdk-config.json, dan accountID* adalah ID Anda. Akun AWS CLI GDK membuat bucket jika tidak ada.

Objek ini berisi informasi berikut:

`bucket`

Nama bucket S3 yang digunakan untuk meng-host artefak komponen.

`region`

Wilayah AWSTempat CLI GDK menerbitkan komponen ini.

Properti ini bersifat opsional jika Anda menggunakan GDK CLI v1.3.0 atau yang lebih baru.

`options`

(Opsional) Opsi konfigurasi tambahan yang digunakan selama pembuatan versi komponen.

Fitur ini tersedia untuk GDK CLI v1.2.0 dan yang lebih baru.

`file_upload_args`

Struktur JSON yang berisi argumen yang dikirim ke Amazon S3 saat mengunggah file ke bucket, seperti metadata dan mekanisme enkripsi. Untuk daftar argumen yang diizinkan, lihat `S3Transfer`kelas dalam dokumentasi Boto3. .

`test-e2e`

(Opsional) Konfigurasi yang akan digunakan selama end-to-end pengujian komponen. Fitur ini tersedia untuk GDK CLI v1.3.0 dan yang lebih baru.

`build`

`build_system`— Sistem build yang akan digunakan. Opsi default adalah`maven`. Pilih dari salah satu pilihan berikut:

- `maven`— Menjalankan `mvn package` perintah untuk membangun modul pengujian. Pilih opsi ini untuk membangun modul pengujian yang menggunakan Maven.
- `gradle`— Menjalankan `gradle build` perintah untuk membangun modul pengujian. Pilih opsi ini untuk modul pengujian yang menggunakan Gradle.

`gtf_version`

(Opsional) Versi Greengrass Testing Framework (GTF) untuk digunakan sebagai dependensi modul pengujian saat Anda menginisialisasi proyek GDK end-to-end dengan GTF. Nilai ini harus menjadi salah satu versi GTF dari rilis. Defaultnya adalah GTF versi 1.1.0.

`gtf_options`

(Opsional) Opsi konfigurasi tambahan yang digunakan selama end-to-end pengujian komponen.

Daftar berikut mencakup opsi yang dapat Anda gunakan dengan GTF versi 1.1.0.

- `additional-plugins`- (Opsional) Plugin Mentimun Tambahan
- `aws-region`— Menargetkan titik akhir regional tertentu untuk AWS layanan. Default untuk apa yang ditemukan SDKAWS.
- `credentials-path`— Jalur kredensial AWS profil opsional. Default untuk kredensil yang ditemukan di lingkungan host.
- `credentials-path-rotation`— Durasi rotasi opsional untuk AWS kredensil. Default hingga 15 menit atau. `PT15M`
- `csr-path`— Jalur untuk CSR yang menggunakan sertifikat perangkat yang akan dihasilkan.
- `device-mode`— Perangkat target yang sedang diuji. Default ke perangkat lokal.
- `env-stage`— Menargetkan lingkungan penyebaran Greengrass. Default untuk produksi.
- `existing-device-cert-arn`— Arn dari sertifikat yang ada yang ingin Anda gunakan sebagai sertifikat perangkat untuk Greengrass.
- `feature-path`— File atau direktori yang berisi file fitur tambahan. Default adalah tidak ada file fitur tambahan yang digunakan.
- `gg-cli-version`— Mengganti versi CLI Greengrass. Default ke nilai yang ditemukan di. `ggc.version`
- `gg-component-bucket`— Nama ember Amazon S3 yang ada yang menampung komponen Greengrass.
- `gg-component-overrides`— Daftar penggantian komponen Greengrass.
- `gg-persist`— Daftar elemen pengujian untuk bertahan setelah uji coba. Perilaku default adalah tidak mempertahankan apa pun. Nilai yang diterima adalah:`aws.resources`,`installed.software`, dan`generated.files`.

- `gg-runtime`— Daftar nilai untuk mempengaruhi bagaimana tes berinteraksi dengan sumber daya pengujian. Nilai-nilai ini menggantikan parameter. `gg.persist` Jika default kosong, itu mengasumsikan semua sumber daya pengujian dikendalikan oleh kasus uji, termasuk runtime Greengrass yang diinstal. Nilai yang diterima adalah:`aws.resources`,`installed.software`, dan`generated.files`.
- `ggc-archive`— Jalur menuju komponen inti Greengrass yang diarsipkan.
- `ggc-install-root`— Direktori untuk menginstal komponen inti Greengrass. Default ke test.temp.path dan test run folder.
- `ggc-log-level`— Atur level log nukleus Greengrass untuk uji coba. Defaultnya adalah “INFO”.
- `ggc-tes-rolename`— Peran IAM yang akan diasumsikan AWS IoT Greengrass Core untuk mengakses AWS layanan. Jika peran dengan nama yang diberikan tidak ada maka akan dibuat dan kebijakan akses default.
- `ggc-trusted-plugins`— Daftar koma terpisah dari jalur (pada host) dari plugin tepercaya yang perlu ditambahkan ke Greengrass. Untuk menyediakan jalur pada DUT itu sendiri, awali jalur dengan 'dut: '
- `ggc-user-name`— Nilai user:group POSIXuser untuk inti Greengrass. Default ke nama pengguna saat ini yang masuk.
- `ggc-version`— Mengganti versi komponen inti Greengrass yang sedang berjalan. Default ke nilai yang ditemukan di ggc.archive.
- `log-level`— Tingkat log uji coba. Default ke “INFO”.
- `parallel-config`— Set indeks batch dan jumlah batch sebagai JSON String. Nilai default indeks batch adalah 0 dan jumlah batch adalah 1.
- `proxy-url`— Konfigurasikan semua tes untuk merutekan lalu lintas melalui URL ini.
- `tags`— Hanya jalankan tag fitur. Dapat berpotongan dengan '&'
- `test-id-prefix`- Awalan umum diterapkan untuk semua sumber daya pengujian tertentu termasuk nama AWS sumber daya dan tag. Default adalah awalan “gg”.
- `test-log-path`— Direktori yang akan berisi hasil dari seluruh uji coba. Default ke “TestResults”.
- `test-results-json`— Tandai untuk menentukan apakah laporan Cucumber JSON yang dihasilkan ditulis ke disk. Default ke true.
- `test-results-log`— Tandai untuk menentukan apakah output konsol dihasilkan ditulis ke disk. Default ke false.

- `test-results-xml`- Tandai untuk menentukan apakah laporan XMLJunit yang dihasilkan dihasilkan ditulis ke disk. Default ke true.
- `test-temp-path`— Direktori untuk menghasilkan artefak uji lokal. Default ke direktori temp acak diawali dengan gg-testing.
- `timeout-multiplier`— Pengganda disediakan untuk semua batas waktu pengujian. Defaultnya adalah 1.0.

Contoh file konfigurasi GDK CLI

Anda dapat mereferensikan contoh file konfigurasi CLI GDK berikut untuk membantu Anda mengonfigurasi lingkungan komponen Greengrass.

Halo Dunia (Python)

File konfigurasi CLI GDK berikut mendukung komponen Hello World yang menjalankan skrip Python. File konfigurasi ini menggunakan sistem `zip` build untuk mengemas skrip Python komponen ke dalam file ZIP yang diunggah CLI GDK sebagai artefak.

```
{
  "component": {
    "com.example.PythonHelloWorld": {
      "author": "Amazon",
      "version": "NEXT_PATCH",
      "build": {
        "build_system" : "zip",
        "options": {
           "excludes": [".*"]
        }
      },
      "publish": {
        "bucket": "greengrass-component-artifacts",
        "region": "us-west-2",
        "options": {
           "file_upload_args": {
              "Metadata": {
                 "some-key": "some-value"
              }
           }
        }
      }
    },
```

```
  "test-e2e":{
    "build":{
        "build_system": "maven"
    },
    "gtf_version": "1.1.0",
    "gtf_options": {
         "tags": "Sample"
     }
  },
  "gdk_version": "1.6.1"
  }
}
```

Hello World (Jawa)

File konfigurasi CLI GDK berikut mendukung komponen Hello World yang menjalankan aplikasi Java. File konfigurasi ini menggunakan sistem `maven` build untuk mengemas kode sumber Java komponen ke dalam file JAR yang diunggah CLI GDK sebagai artefak.

```
{
  "component": {
    "com.example.JavaHelloWorld": {
      "author": "Amazon",
      "version": "NEXT_PATCH",
      "build": {
        "build_system" : "maven"
      },
      "publish": {
        "bucket": "greengrass-component-artifacts",
        "region": "us-west-2",
        "options": {
           "file_upload_args": {
              "Metadata": {
                 "some-key": "some-value"
              }
           }
        }
      }
  },
  "test-e2e":{
    "build":{
        "build_system": "maven"
    },
```

```
    "gtf_version": "1.1.0",
    "gtf_options": {
         "tags": "Sample"
     }
  },
  "gdk_version": "1.6.1"
  }
}
```

Komponen komunitas

Beberapa komponen komunitas dalam Katalog Perangkat Lunak Greengrass menggunakan CLI GDK. Anda dapat menjelajahi file konfigurasi CLI GDK di repositori komponen ini.

Untuk melihat file konfigurasi CLI GDK komponen komunitas

1. Jalankan perintah berikut untuk membuat daftar komponen komunitas yang menggunakan CLI GDK.

   ```
   gdk component list --repository
   ```

   Respons mencantumkan nama GitHub repositori untuk setiap komponen komunitas yang menggunakan CLI GDK. Setiap repositori ada di organisasi. `awslabs`

   ```
   [2022-02-22 17:27:31] INFO - Listing all the available component repositories from
    Greengrass Software Catalog.
   [2022-02-22 17:27:31] INFO - Found '6' component repositories to display.
   1. aws-greengrass-labs-database-influxdb
   2. aws-greengrass-labs-telemetry-influxdbpublisher
   3. aws-greengrass-labs-dashboard-grafana
   4. aws-greengrass-labs-dashboard-influxdb-grafana
   5. aws-greengrass-labs-local-web-server
   6. aws-greengrass-labs-lookoutvision-gstreamer
   ```

2. Buka GitHub repositori komponen komunitas di URL berikut. Ganti *`community-component-name`*dengan nama komponen komunitas dari langkah sebelumnya.

   ```
   https://github.com/awslabs/community-component-name
   ```

Antarmuka Baris Perintah Greengrass

Greengrass Command Line Interface (CLI) memungkinkan Anda berinteraksi dengan AWS IoT Greengrass Inti pada perangkat Anda untuk mengembangkan komponen dan men-debug masalah secara lokal. Misalnya, Anda dapat menggunakan Greengrass CLI untuk membuat deployment lokal dan me-restart komponen pada perangkat inti.

Deploy Komponen Greengrass CLI (`aws.greengrass.Cli`) untuk menginstal Greengrass CLI pada perangkat inti Anda.

> **Important**
>
> Kami menyarankan Anda menggunakan komponen ini hanya di lingkungan pengembangan, bukan lingkungan produksi. Komponen ini menyediakan akses ke informasi dan operasi yang biasanya tidak Anda perlukan di lingkungan produksi. Ikuti prinsip hak istimewa paling sedikit dengan menerapkan komponen ini hanya ke perangkat inti di mana Anda membutuhkannya.

Topik

Instal Greengrass CLI

Anda dapat menginstal Greengrass CLI dengan salah satu cara berikut:

- Gunakan `--deploy-dev-tools` argumen saat pertama kali menyiapkan perangkat lunak AWS IoT Greengrass Core di perangkat Anda. Anda juga harus menentukan `--provision true` untuk menerapkan argumen ini.
- Deploy komponen Greengrass CLI (`aws.greengrass.Cli`) di perangkat Anda.

Bagian ini menjelaskan langkah-langkah untuk men-deploy komponen Greengrass CLI. Untuk informasi tentang menginstal Greengrass CLI selama pengaturan awal, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2.

Prasyarat

Untuk menerapkan komponen CLI Greengrass, Anda harus memenuhi persyaratan berikut:

- AWS IoT Greengrass Perangkat lunak inti diinstal dan dikonfigurasi pada perangkat inti Anda. Untuk informasi selengkapnya, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2.
- Untuk menggunakan AWS CLI untuk menyebarkan CLI Greengrass, Anda harus telah menginstal dan mengkonfigurasi. AWS CLI Untuk informasi lebih lanjut, lihat Mengonfigurasi AWS CLI di Panduan Pengguna AWS Command Line Interface .
- Anda harus diberi wewenang untuk menggunakan CLI Greengrass untuk berinteraksi dengan perangkat lunak Core. AWS IoT Greengrass Lakukan salah satu langkah berikut untuk menggunakan Greengrass CLI:
  - Gunakan pengguna sistem yang menjalankan perangkat lunak AWS IoT Greengrass Core.
  - Gunakan pengguna dengan izin root atau adminstratif. Pada perangkat inti Linux, Anda dapat menggunakan `sudo` untuk mendapatkan izin root.
  - Gunakan pengguna sistem yang berada dalam grup yang Anda tentukan dalam parameter `AuthorizedPosixGroups` atau `AuthorizedWindowsGroups` konfigurasi saat Anda menerapkan komponen. Untuk informasi selengkapnya, lihat konfigurasi komponen CLI Greengrass.

Deploy komponen Greengrass CLI

Selesaikan langkah-langkah berikut untuk men-deploy komponen Greengrass CLI ke perangkat inti Anda:

Untuk men-deploy komponen Greengrass CLI (konsol)

1. Masuk ke konsol AWS IoT Greengrass.
2. Dalam menu navigasi, pilih Komponen.
3. Pada halaman Komponen, pada tab Komponen publik, pilih `aws.greengrass.Cli`.
4. Pada halaman aws.greengrass.Cli pilih Deploy.
5. Dari Tambahkan ke deployment, pilih Buat deployment baru.
6. Pada halaman Tentukan target, di bawah Target deployment, di daftar Nama target, pilih grup Greengrass yang ingin Anda deploy, dan pilih Selanjutnya.
7. Pada halaman Pilih komponen, verifikasi bahwa komponen aws.greengrass.Cli tersebut dipilih, dan pilih Selanjutnya.
8. Pada halaman Konfigurasikan komponen, simpan pengaturan konfigurasi default, dan pilih Selanjutnya.

9. Pada halaman Konfigurasikan pengaturan lanjutan, simpan pengaturan konfigurasi default tersebut, dan pilih Selanjutnya.
10. Di halaman Tinjauan, klik Deploy.

Untuk men-deploy komponen Greengrass CLI (AWS CLI)

1. Di perangkat Anda, buatlah file `deployment.json` untuk menentukan konfigurasi deployment untuk komponen Greengrass CLI. File ini akan terlihat seperti berikut:

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.Cli": {
      "componentVersion": "2.12.6",
      "configurationUpdate": {
        "merge": "{\"AuthorizedPosixGroups\":\"<group1>,<group2>,...,<groupN>\",
\"AuthorizedWindowsGroups\":\"<group1>,<group2>,...,<groupN>\"}"
      }
    }
  }
}
```

   - Di kolom `target`, ganti *`targetArn`* dengan Amazon Resource Name (ARN) dari grup objek atau objek yang ditargetkan untuk deployment tersebut, dalam format berikut:
     - Objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Grup objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
   - Di objek komponen `aws.greengrass.Cli`, tentukan nilai-nilai sebagai berikut:

     `version`

     Versi komponen Greengrass CLI.

     `configurationUpdate.AuthorizedPosixGroups`

     (Opsional) String yang berisi daftar kelompok sistem yang dipisahkan dengan koma. Anda mengizinkan grup sistem ini untuk menggunakan CLI Greengrass untuk berinteraksi dengan perangkat lunak Inti. AWS IoT Greengrass Anda dapat menentukan nama grup atau ID grup. Misalnya, `group1,1002,group3` mengotorisasi tiga grup sistem (`group1`, `1002`, dan `group3`) untuk menggunakan Greengrass CLI.

Jika Anda tidak menentukan grup apa pun untuk diotorisasi, Anda dapat menggunakan Greengrass CLI sebagai `sudo` pengguna root () atau sebagai pengguna sistem yang menjalankan perangkat lunak Core. AWS IoT Greengrass

`configurationUpdate.AuthorizedWindowsGroups`

(Opsional) String yang berisi daftar kelompok sistem yang dipisahkan dengan koma. Anda mengizinkan grup sistem ini untuk menggunakan CLI Greengrass untuk berinteraksi dengan perangkat lunak Inti. AWS IoT Greengrass Anda dapat menentukan nama grup atau ID grup. Misalnya, `group1,1002,group3` mengotorisasi tiga grup sistem (`group1`, `1002`, dan `group3`) untuk menggunakan Greengrass CLI.

Jika Anda tidak menentukan grup apa pun untuk diotorisasi, Anda dapat menggunakan CLI Greengrass sebagai administrator atau sebagai pengguna sistem yang menjalankan perangkat lunak Core. AWS IoT Greengrass

2. Jalankan perintah berikut untuk men-deploy komponen Greengrass CLI pada perangkat:

```
$ aws greengrassv2 create-deployment --cli-input-json file://path/
to/deployment.json
```

Selama instalasi, komponen menambahkan link simbolik ke `greengrass-cli` dalam folder */greengrass/v2*/`bin` pada perangkat Anda, dan Anda menjalankan Greengrass CLI dari jalur ini. Untuk menjalankan Greengrass CLI tanpa path absolut, tambahkan folder */greengrass/v2*/`bin` Anda ke variabel PATH Anda. Untuk memverifikasi instalasi Greengrass CLI, jalankan perintah berikut:

Linux or Unix

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

Windows

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

Anda akan melihat output berikut:

```
Usage: greengrass-cli [-hV] [--ggcRootPath=<ggcRootPath>] [COMMAND]
```

```
Greengrass command line interface

      --ggcRootPath=<ggcRootPath>
                  The AWS IoT Greengrass V2 root directory.
  -h, --help      Show this help message and exit.
  -V, --version   Print version information and exit.
Commands:
  help                Show help information for a command.
  component           Retrieve component information and stop or restart
                        components.
  deployment          Create local deployments and retrieve deployment status.
  logs                Analyze Greengrass logs.
  get-debug-password  Generate a password for use with the HTTP debug view
                        component.
```

Jika `greengrass-cli` tidak ditemukan, deployment mungkin gagal untuk menginstal Greengrass CLI. Untuk informasi selengkapnya, lihat Pemecahan masalah AWS IoT Greengrass V2.

Perintah Greengrass CLI

Greengrass CLI menyediakan antarmuka baris perintah untuk berinteraksi secara lokal dengan AWS IoT Greengrass perangkat inti. Perintah Greengrass CLI menggunakan format berikut.

```
$ greengrass-cli <command> <subcommand> [arguments]
```

Secara default, file yang `greengrass-cli` dapat dieksekusi di */greengrass/v2*`/bin/` folder berinteraksi dengan versi perangkat lunak AWS IoT Greengrass Core yang berjalan di folder. */greengrass/v2* Jika Anda memanggil executable yang tidak ditempatkan di lokasi ini, atau jika Anda ingin berinteraksi dengan AWS IoT Greengrass Perangkat lunak inti di lokasi yang berbeda, Anda harus menggunakan salah satu metode berikut untuk secara tegas menentukan jalur akar AWS IoT Greengrass Perangkat lunak inti yang ingin Anda jalin interaksi dengannya:

- Atur `GGC_ROOT_PATH` variabel lingkungan ke */greengrass/v2*.
- Tambahkan `--ggcRootPath` */greengrass/v2* argumen ke perintah Anda seperti yang ditunjukkan pada contoh berikut.

  ```
  greengrass-cli --ggcRootPath /greengrass/v2 <command> <subcommand> [arguments]
  ```

Anda dapat menggunakan argumen berikut dengan perintah apa pun:

- Gunakan `--help` untuk informasi tentang perintah Greengrass CLI tertentu.
- Gunakan `--version` untuk informasi tentang versi Greengrass CLI.

Bagian ini menjelaskan perintah Greengrass CLI dan memberikan contoh untuk perintah ini. Sinopsis untuk setiap perintah menunjukkan argumen dan penggunaannya. Argumen opsional ditampilkan dalam tanda kurung siku.

Perintah yang tersedia

- komponen
- deployment
- log
- get-debug-password

komponen

Gunakan perintah `component` untuk berinteraksi dengan komponen lokal pada perangkat inti Anda.

Subperintah

- detail
- Daftar
- restart
- berhenti

detail

Ambil versi, status, dan konfigurasi dari satu komponen.

Sinopsis

```
greengrass-cli component details --name <component-name>
```

Argumen

`--name`, `-n`. Nama komponen.

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ sudo greengrass-cli component details --name MyComponent

Component Name: MyComponent
Version: 1.0.0
State: RUNNING
Configuration: null
```

Daftar

Ambil nama, versi, status, dan konfigurasi setiap komponen yang diinstal pada perangkat.

Sinopsis

```
greengrass-cli component list
```

Argumen

Tidak ada

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ sudo greengrass-cli component list

Components currently running in Greengrass:
Component Name: FleetStatusService
Version: 0.0.0
State: RUNNING
Configuration: {"periodicUpdateIntervalSec":86400.0}
Component Name: UpdateSystemPolicyService
Version: 0.0.0
State: RUNNING
Configuration: null
Component Name: aws.greengrass.Nucleus
Version: 2.0.0
State: FINISHED
```

```
Configuration: {"awsRegion":"region","runWithDefault":
{"posixUser":"ggc_user:ggc_group"},"telemetry":{}}
Component Name: DeploymentService
Version: 0.0.0
State: RUNNING
Configuration: null
Component Name: TelemetryAgent
Version: 0.0.0
State: RUNNING
Configuration: null
Component Name: aws.greengrass.Cli
Version: 2.0.0
State: RUNNING
Configuration: {"AuthorizedPosixGroups":"ggc_user"}
```

restart

Mulai ulang komponen.

Sinopsis

```
greengrass-cli component restart --names <component-name>,...
```

Argumen

`--names`, `-n`. Nama komponen. Setidaknya satu nama komponen diperlukan. Anda dapat menentukan nama komponen tambahan, yang memisahkan setiap nama dengan koma.

Keluaran

Tidak ada

berhenti

Berhenti menjalankan komponen.

Sinopsis

```
greengrass-cli component stop --names <component-name>,...
```

Argumen

`--names`, `-n`. Nama komponen. Setidaknya satu nama komponen diperlukan. Anda dapat menentukan nama komponen tambahan jika diperlukan, yang memisahkan setiap nama dengan koma.

Keluaran

Tidak ada

deployment

Gunakan perintah `deployment` untuk berinteraksi dengan komponen lokal pada perangkat inti Anda.

Untuk memantau kemajuan penerapan lokal, gunakan `status` subperintah. Anda tidak dapat memantau kemajuan penerapan lokal menggunakan konsol.

Subperintah

- buat
- membatalkan
- daftar
- status

buat

Buat atau perbarui deployment lokal menggunakan resep komponen tertentu, artefak, dan argumen waktu aktif.

Sinopsis

```
greengrass-cli deployment create
    --recipeDir path/to/component/recipe
    [--artifactDir path/to/artifact/folder ]
    [--update-config {component-configuration}]
    [--groupId <thing-group>]
    [--merge "<component-name>=<component-version>"]...
    [--runWith "<component-name>:posixUser=<user-name>[:<group-name>]"]...
    [--systemLimits "{component-system-resource-limits}]"]...
    [--remove <component-name>,...]
    [--failure-handling-policy <policy name[ROLLBACK, DO_NOTHING]>]
```

Argumen

- `--recipeDir`, `-r`. Jalur lengkap ke folder yang berisi file resep komponen.
- `--artifactDir`, `-a`. Jalur lengkap ke folder yang berisi file artefak yang ingin Anda sertakan dalam deployment Anda. Folder artefak harus berisi struktur direktori berikut:

  ```
  /path/to/artifact/folder/<component-name>/<component-version>/<artifacts>
  ```

- `--update-config`, `-c`. Argumen konfigurasi untuk deployment, disediakan sebagai string JSON atau file JSON. String JSON harus dalam format berikut:

  ```
  { \
    "componentName": { \
      "MERGE": {"config-key": "config-value"}, \
      "RESET": ["path/to/reset/"] \
    } \
  }
  ```

  `MERGE` dan `RESET` peka huruf besar-kecil dan harus dalam huruf besar.
- `--groupId`, `-g`. Grup sasaran objek untuk deployment.
- `--merge`, `-m`. Nama dan versi komponen target yang ingin Anda tambahkan atau perbarui. Anda harus memberikan informasi komponen dalam format *<component>*=*<version>*. Gunakan argumen terpisah untuk setiap komponen tambahan yang akan ditentukan. Jika diperlukan, gunakan `--runWith` argumen untuk memberikan`posixUser`,`posixGroup`, dan `windowsUser` informasi untuk menjalankan komponen.
- `--runWith`. The`posixUser`,`posixGroup`, dan `windowsUser` informasi untuk menjalankan komponen generik atau Lambda. Anda harus memberikan informasi ini dalam format *<component>*`:{posixUser|windowsUser}`=*<user>*`[:`*<=posixGroup>*`]`. Misalnya, Anda dapat menentukan **`HelloWorld:posixUser=ggc_user:ggc_group`** atau**`HelloWorld:windowsUser=ggc_user`**. Gunakan argumen terpisah untuk setiap opsi tambahan yang akan ditentukan.

  Untuk informasi selengkapnya, lihat Konfigurasikan pengguna yang menjalankan komponen.
- `--systemLimits`. Batas sumber daya sistem untuk diterapkan pada proses komponen Lambda generik dan non-kontainer pada perangkat inti. Anda dapat mengonfigurasi jumlah maksimum penggunaan CPU dan RAM yang dapat digunakan oleh setiap proses komponen. Tentukan objek JSON serial atau path file ke file JSON. Objek JSON harus memiliki format berikut.

```
{  \
  "componentName": { \
    "cpus": cpuTimeLimit, \
    "memory": memoryLimitInKb \
  } \
}
```

Anda dapat mengonfigurasi batas sumber daya sistem berikut untuk setiap komponen:

- `cpus`— Jumlah maksimum waktu CPU yang dapat digunakan oleh proses komponen ini pada perangkat inti. Total waktu CPU perangkat inti setara dengan jumlah inti CPU perangkat. Misalnya, pada perangkat inti dengan 4 core CPU, Anda dapat mengatur nilai ini 2 untuk membatasi proses komponen ini hingga 50 persen penggunaan setiap inti CPU. Pada perangkat dengan 1 inti CPU, Anda dapat mengatur nilai ini `0.25` untuk membatasi proses komponen ini hingga 25 persen penggunaan CPU. Jika Anda menetapkan nilai ini ke angka yang lebih besar dari jumlah inti CPU, perangkat lunak AWS IoT Greengrass Core tidak membatasi penggunaan CPU komponen.
- `memory`— Jumlah maksimum RAM (dalam kilobyte) yang dapat digunakan oleh proses komponen ini pada perangkat inti.

Untuk informasi selengkapnya, lihat Konfigurasikan batas sumber daya sistem untuk komponen.

Fitur ini tersedia untuk v2.4.0 dan yang lebih baru dari komponen inti Greengrass dan CLI Greengrass pada perangkat inti Linux. AWS IoT Greengrass saat ini tidak mendukung fitur ini di perangkat inti Windows.

- `--remove`. Nama komponen target yang ingin Anda hapus dari deployment lokal. Untuk menghapus komponen yang digabung dari deployment cloud, Anda harus memberikan ID grup dari grup objek target dalam format berikut:

  Greengrass nucleus v2.4.0 and later

  ```
  --remove <component-name> --groupId <group-name>
  ```

  Earlier than v2.4.0

  ```
  --remove <component-name> --groupId thinggroup/<group-name>
  ```

- `--failure-handling-policy`. Mendefinisikan tindakan yang diambil saat penerapan gagal. Ada dua tindakan yang dapat Anda tentukan:
  - `ROLLBACK` –
  - `DO_NOTHING` –

  Fitur ini tersedia untuk v2.11.0 dan yang lebih baru. Inti Greengrass

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ sudo greengrass-cli deployment create \
    --merge MyApp1=1.0.0 \
    --merge MyApp2=1.0.0 --runWith MyApp2:posixUser=ggc_user \
    --remove MyApp3 \
    --recipeDir recipes/ \
    --artifactDir artifacts/

Local deployment has been submitted! Deployment Id: 44d89f46-1a29-4044-
ad89-5151213dfcbc
```

membatalkan

Membatalkan penerapan yang ditentukan.

Sinopsis

```
greengrass-cli deployment cancel
    -i <deployment-id>
```

Argumen

`-i`. Pengenal unik dari penerapan untuk membatalkan. ID penyebaran dikembalikan dalam output `create` perintah.

Output

- Tidak ada

daftar

Ambil status dari 10 deployment lokal terakhir.

Sinopsis

```
greengrass-cli deployment list
```

Argumen

Tidak ada

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini. Tergantung pada status deployment Anda, output menunjukkan salah satu dari nilai status berikut: `IN_PROGRESS`, `SUCCEEDED`, atau `FAILED`.

```
$ sudo greengrass-cli deployment list

44d89f46-1a29-4044-ad89-5151213dfcbc: SUCCEEDED
Created on: 6/27/23 11:05 AM
```

status

Ambil status deployment tertentu.

Sinopsis

```
greengrass-cli deployment status -i <deployment-id>
```

Argumen

`-i`. ID deployment.

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini. Tergantung pada status deployment Anda, output menunjukkan salah satu dari nilai status berikut: `IN_PROGRESS`, `SUCCEEDED`, atau `FAILED`.

```
$ sudo greengrass-cli deployment status -i 44d89f46-1a29-4044-ad89-5151213dfcbc

44d89f46-1a29-4044-ad89-5151213dfcbc: FAILED
Created on: 6/27/23 11:05 AM
```

```
Detailed Status: <Detailed deployment status>
Deployment Error Stack: List of error codes
Deployment Error Types: List of error types
Failure Cause: Cause
```

log

Gunakan perintah `logs` untuk menganalisis log Greengrass pada perangkat inti Anda.

Subperintah

- memperoleh
- list-keywords
- list-log-files

memperoleh

Kumpulkan, filter, dan visualisasikan file log Greengrass. Perintah ini hanya mendukung file log berformat JSON. Anda dapat menentukan format logging dalam konfigurasi nukleus.

Sinopsis

```
greengrass-cli logs get
    [--log-dir path/to/a/log/folder]
    [--log-file path/to/a/log/file]
    [--follow true | false ]
    [--filter <filter> ]
    [--time-window <start-time>,<end-time> ]
    [--verbose ]
    [--no-color ]
    [--before <value> ]
    [--after <value> ]
    [--syslog ]
    [--max-long-queue-size <value> ]
```

Argumen

- `--log-dir`,`-ld`. Jalur ke direktori untuk memeriksa file log, seperti***/greengrass/v2/*****logs**. Jangan gunakan dengan `--syslog`. Gunakan argumen terpisah untuk setiap direktori

tambahan yang akan ditentukan. Anda harus menggunakan setidaknya salah satu dari `--log-dir` atau `--log-file`. Anda juga dapat menggunakan kedua argumen dalam satu perintah.

- `--log-file`, `-lf`. Jalur ke direktori log yang ingin Anda gunakan. Gunakan argumen terpisah untuk setiap direktori tambahan yang akan ditentukan. Anda harus menggunakan setidaknya salah satu dari `--log-dir` atau `--log-file`. Anda juga dapat menggunakan kedua argumen dalam satu perintah.
- `--follow`, `-fol`. Tampilkan pembaruan log saat terjadi. Greengrass CLI terus berjalan dan membaca dari log tertentu. Jika Anda menentukan jendela waktu, Greengrass CLI berhenti memantau log setelah semua jendela waktu berakhir.
- `--filter`, `-f`. Kata kunci, ekspresi reguler, atau pasangan kunci-nilai untuk digunakan sebagai filter. Berikan nilai ini sebagai string, ekspresi reguler, atau sebagai pasangan kunci-nilai. Gunakan argumen terpisah untuk setiap filter tambahan yang akan ditentukan.

  Ketika dievaluasi, beberapa filter yang ditentukan dalam argumen tunggal dipisahkan oleh operator OR, dan filter yang ditentukan dalam argumen tambahan digabungkan dengan operator AND. Misalnya, jika perintah Anda mencakup `--filter "installed" --filter "name=alpha,name=beta"`, maka Greengrass CLI akan memfilter dan menampilkan pesan log yang berisi baik kata kunci `installed` maupun kunci `name` yang memiliki nilai `alpha` atau `beta`.
- `--time-window`, `-t`. Jendela waktu untuk menampilkan informasi log. Anda dapat menggunakan kedua stempel waktu yang tepat dan offset relatif. Anda harus memberikan informasi ini dalam format *`<begin-time>`*`,`*`<end-time>`*. Jika Anda tidak menentukan baik waktu mulai maupun waktu akhir, nilai untuk opsi default ke tanggal dan waktu sistem saat ini. Gunakan argumen terpisah untuk setiap jendela waktu tambahan yang akan ditentukan.

  Greengrass CLI mendukung format berikut untuk stempel waktu:

  - `yyyy-MM-DD`, misalnya, `2020-06-30`. Waktu default adalah 00:00:00 ketika Anda menggunakan format ini.

    `yyyyMMDD`, misalnya, `20200630`. Waktu default adalah 00:00:00 ketika Anda menggunakan format ini.

    `HH:mm:ss`, misalnya, `15:30:45`. Tanggal default adalah tanggal sistem saat ini ketika Anda menggunakan format ini.

    `HH:mm:ssSSS`, misalnya, `15:30:45`. Tanggal default adalah tanggal sistem saat ini ketika Anda menggunakan format ini.

`YYYY-MM-DD'T'HH:mm:ss'Z'`, misalnya, `2020-06-30T15:30:45Z`.

`YYYY-MM-DD'T'HH:mm:ss`, misalnya, `2020-06-30T15:30:45`.

`yyyy-MM-dd'T'HH:mm:ss.SSS`, misalnya, `2020-06-30T15:30:45.250`.

Offset relatif menentukan jangka waktu offset dari waktu sistem saat ini. Greengrass CLI mendukung format berikut untuk offset relatif: `+|-[<value>h|hr|hours][value m|min|minutes][value]s|sec|seconds`.

Misalnya, argumen berikut untuk menentukan jendela waktu antara 1 jam dan 2 jam 15 menit sebelum waktu saat ini adalah `--time-window -2h15min,-1hr`.

- `--verbose`. Tampilkan semua kolom dari pesan log. Jangan gunakan dengan `--syslog`.
- `--no-color`, `-nc`. Hapus kode warna. Default pengkodean warna untuk pesan log menggunakan teks merah tebal. Mendukung hanya terminal seperti UNIX karena menggunakan ANSI escape sequence.
- `--before`, `-b`. Jumlah baris yang akan ditunjukkan mendahului entri log yang cocok. Default-nya adalah 0.
- `--after`, `-a`. Jumlah baris yang akan ditunjukkan mengikuti entri log yang cocok. Default-nya adalah 0.
- `--syslog`. Proses semua file log menggunakan protokol syslog yang ditentukan oleh RFC3164. Jangan gunakan dengan `--log-dir` dan `--verbose`. Protokol syslog menggunakan format berikut: `"<$Priority>$Timestamp $Host $Logger ($Class): $Message"`. Jika Anda tidak menentukan berkas log, Greengrass CLI akan membaca pesan log dari lokasi-lokasi berikut: `/var/log/messages`, `/var/log/syslog`, atau `/var/log/system.log`.

  AWS IoT Greengrasssaat ini tidak mendukung fitur ini di perangkat inti Windows.
- `--max-log-queue-size`, `-m`. Jumlah maksimum entri log yang akan dialokasikan ke memori. Gunakan opsi ini untuk mengoptimalkan penggunaan memori. Default adalah 100.

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ sudo greengrass-cli logs get --verbose \
    --log-file /greengrass/v2/logs/greengrass.log \
    --filter deployment,serviceName=DeploymentService \
```

```
    --filter level=INFO \
    --time-window 2020-12-08T01:11:17,2020-12-08T01:11:22

2020-12-08T01:11:17.615Z [INFO] (pool-2-thread-14)
 com.aws.greengrass.deployment.DeploymentService: Current deployment finished.
 {DeploymentId=44d89f46-1a29-4044-ad89-5151213dfcbc, serviceName=DeploymentService,
 currentState=RUNNING}
2020-12-08T01:11:17.675Z [INFO] (pool-2-thread-14)
 com.aws.greengrass.deployment.IotJobsHelper: Updating status of persisted
 deployment. {Status=SUCCEEDED, StatusDetails={detailed-deployment-
status=SUCCESSFUL}, ThingName=MyThing, JobId=22d89f46-1a29-4044-ad89-5151213dfcbc
```

list-keywords

Tampilkan kata kunci yang disarankan yang dapat Anda gunakan untuk memfilter berkas log.

Sinopsis

```
greengrass-cli logs list-keywords [arguments]
```

Argumen

Tidak ada

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ sudo greengrass-cli logs list-keywords

Here is a list of suggested keywords for Greengrass log:
level=$str
thread=$str
loggerName=$str
eventType=$str
serviceName=$str
error=$str
```

```
$ sudo greengrass-cli logs list-keywords --syslog

Here is a list of suggested keywords for syslog:
priority=$int
```

```
host=$str
logger=$str
class=$str
```

list-log-files

Tampilkan file log yang terletak di direktori tertentu.

Sinopsis

```
greengrass-cli logs list-log-files [arguments]
```

Argumen

`--log-dir`, `-ld`. Jalur ke direktori untuk memeriksa file log.

Keluaran

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ sudo greengrass-cli logs list-log-files -ld /greengrass/v2/logs/

/greengrass/v2/logs/aws.greengrass.Nucleus.log
/greengrass/v2/logs/main.log
/greengrass/v2/logs/greengrass.log
Total 3 files found.
```

get-debug-password

Menggunakan perintah `get-debug-password` untuk mencetak sandi yang dihasilkan secara acak untuk komponen konsol debug lokal (`aws.greengrass.LocalDebugConsole`). Kata sandi kedaluwarsa 8 jam setelah dibuat.

Ringkasan

```
greengrass-cli get-debug-password
```

Pendapat

Tidak ada

Output

Contoh berikut menunjukkan output yang dihasilkan ketika Anda menjalankan perintah ini.

```
$ sudo greengrass-cli get-debug-password

Username: debug
Password: bEDp3MOHdj8ou2w5de_sCBI2XAaguy3a8XxREXAMPLE
Password expires at: 2021-04-01T17:01:43.921999931-07:00
The local debug console is configured to use TLS security. The certificate is self-
signed so you will need to bypass your web browser's security warnings to open the
 console.
Before you bypass the security warning, verify that the certificate fingerprint
 matches the following fingerprints.
SHA-256: 15 0B 2C E2 54 8B 22 DE 08 46 54 8A B1 2B 25 DE FB 02 7D 01 4E 4A 56 67 96
 DA A6 CC B1 D2 C4 1B
SHA-1: BC 3E 16 04 D3 80 70 DA E0 47 25 F9 90 FA D6 02 80 3E B5 C1
```

Gunakan Kerangka AWS IoT Greengrass Pengujian

Greengrass Testing Framework (GTF) adalah kumpulan blok bangunan yang mendukung otomatisasi dari perspektif pelanggan. end-to-end GTF menggunakan Cucumber sebagai driver fitur. AWS IoT Greengrass menggunakan blok bangunan yang sama untuk memenuhi syarat perubahan perangkat lunak pada berbagai perangkat. Untuk informasi lebih lanjut, lihat Greengrass Testing Framework di Github.

GTF diimplementasikan menggunakan Cucumber, alat yang digunakan untuk menjalankan tes otomatis, untuk mendorong Behavior-Driven Development (BDD) komponen. Di Mentimun, fitur-fitur sistem ini diuraikan dalam jenis file khusus yang disebut`feature`. Setiap fitur dijelaskan dalam format yang dapat dibaca manusia yang disebut skenario yang merupakan spesifikasi yang dapat dikonversi menjadi pengujian otomatis. Setiap skenario diuraikan sebagai serangkaian langkah yang menentukan interaksi dan hasil dari sistem ini yang diuji menggunakan bahasa khusus domain yang disebut Gherkin. Langkah Gherkin ditautkan ke kode pemrograman menggunakan metode yang disebut definisi langkah yang menghubungkan spesifikasi dengan keras ke aliran pengujian. Definisi langkah dalam GTF diimplementasikan dengan Java.

Topik

- Cara kerjanya
- Changelog

- Opsi konfigurasi Kerangka Pengujian Greengrass
- Tutorial: Jalankan end-to-end tes menggunakan Greengrass Testing Framework dan Greengrass Development Kit
- Tutorial: Gunakan tes kepercayaan diri dari rangkaian tes kepercayaan

Cara kerjanya

AWS IoT Greengrass mendistribusikan GTF sebagai JAR mandiri yang terdiri dari beberapa modul Java. Untuk menggunakan GTF untuk end-to-end pengujian komponen, Anda harus mengimplementasikan pengujian dalam proyek Java. Menambahkan JAR standar pengujian sebagai dependensi dalam proyek Java Anda memungkinkan Anda untuk menggunakan fungsionalitas GTF yang ada dan memperluasnya dengan menulis kasus uji kustom Anda sendiri. Untuk menjalankan kasus uji kustom, Anda dapat membangun proyek Java Anda dan menjalankan JAR target dengan opsi konfigurasi yang dijelaskan diOpsi konfigurasi Kerangka Pengujian Greengrass.

JAR mandiri GTF

Greengrass menggunakan Cloudfront sebagai repositori Maven untuk meng-host berbagai versi JAR mandiri GTF. Untuk daftar lengkap versi GTF, lihat rilis GTF.

JAR mandiri GTF mencakup modul-modul berikut. Tidak terbatas hanya pada modul-modul ini. Anda dapat memilih dan memilih masing-masing dependensi ini secara terpisah dalam proyek Anda atau menyertakan semuanya sekaligus dengan file JAR mandiri pengujian.

- `aws-greengrass-testing-resources`: Modul ini menyediakan abstraksi untuk mengelola siklus hidup AWS sumber daya selama pengujian. Anda dapat menggunakan ini untuk menentukan AWS sumber daya kustom Anda menggunakan `ResourceSpec` abstraksi sehingga GTF dapat menangani pembuatan dan penghapusan sumber daya tersebut untuk Anda.
- `aws-greengrass-testing-platform`: Modul ini menyediakan abstraksi tingkat platform untuk perangkat yang diuji selama siklus hidup pengujian. Ini berisi API yang digunakan untuk berinteraksi dengan OS independen dari platform dan dapat digunakan untuk mensimulasikan perintah yang berjalan di shell perangkat.
- `aws-greengrass-testing-components`: Modul ini terdiri dari komponen sampel yang digunakan untuk menguji fitur inti Greengrass seperti penerapan, IPC, dan fitur lainnya.
- `aws-greengrass-testing-features`: Modul ini terdiri dari langkah-langkah umum yang dapat digunakan kembali dan definisinya yang digunakan untuk pengujian di dalam lingkungan Greengrass.

Topik

Changelog

Tabel berikut menjelaskan perubahan di setiap versi GTF. Untuk informasi selengkapnya, lihat halaman Rilis GTF di GitHub.

Versi	Perubahan
1.2.0	Fitur baru • Menambahkan langkah-langkah terkait jaringan untuk mengonfigurasi MQTT dan konektivitas jaringan internet selama pengujian. • Menambahkan langkah-langkah metrik sistem untuk memantau penggunaan RAM dan CPU perangkat. Perbaikan bug dan peningkatan • Langkah penerapan lokal Greengrass CLI mencoba lagi hingga berhasil. • Tes dengan anggun menghentikan inti Greengrass alih-alih membunuhn ya. • Menambahkan peningkatan di mana GTF melakukan polling titik akhir AWS IoT Credentials hingga kredensyal dapat diambil untuk hal dan alias peran. • Memperbaiki artefak dan direktori resep yang hilang. Versi ini juga memperbaiki versi komponen yang hilang. • Memperbaiki masalah saat GTF gagal selama pembersihan gambar docker jika image docker tidak ada. • Menambahkan kata kunci CURRENT sebagai versi komponen.

Versi	Perubahan
1.1.0	Fitur baru • Menambahkan kemampuan untuk menginstal komponen kustom dengan konfigurasi. Ini membutuhkan resep untuk komponen khusus. • Menambahkan kemampuan untuk memperbarui penerapan lokal dengan konfigurasi khusus. Perbaikan bug dan peningkatan • Memperbaiki masalah inkonsistensi versi GTF konteks log.
1.0.0	Versi awal.

Opsi konfigurasi Kerangka Pengujian Greengrass

Opsi konfigurasi GTF

Greengrass Testing Framework (GTF) memungkinkan Anda untuk mengkonfigurasi parameter tertentu selama peluncuran end-to-end proses pengujian untuk mengatur aliran uji. Anda dapat menentukan opsi konfigurasi ini sebagai argumen CLI untuk JAR mandiri GTF.

GTF versi 1.1.0 dan yang lebih baru menyediakan opsi konfigurasi berikut.

- `additional-plugins`- (Opsional) Plugin Mentimun Tambahan
- `aws-region`— Menargetkan titik akhir regional tertentu untukAWSlayanan. Default untuk apaAWSSDK menemukan.
- `credentials-path`- OpsionalAWSjalur kredensi profil. Default untuk kredensil yang ditemukan di lingkungan host.
- `credentials-path-rotation`— Durasi rotasi opsional untukAWSkredensil. Default hingga 15 menit atau`PT15M`.
- `csr-path`— Jalur untuk CSR yang menggunakan sertifikat perangkat yang akan dihasilkan.
- `device-mode`— Perangkat target yang sedang diuji. Default ke perangkat lokal.
- `env-stage`— Menargetkan lingkungan penyebaran Greengrass. Default untuk produksi.
- `existing-device-cert-arn`— Arn dari sertifikat yang ada yang ingin Anda gunakan sebagai sertifikat perangkat untuk Greengrass.

- `feature-path`— File atau direktori yang berisi file fitur tambahan. Default adalah tidak ada file fitur tambahan yang digunakan.
- `gg-cli-version`— Mengganti versi Greengrass CLI. Default ke nilai yang ditemukan di`ggc.version`.
- `gg-component-bucket`— Nama bucket Amazon S3 yang ada yang menampung komponen Greengrass.
- `gg-component-overrides`— Daftar penggantian komponen Greengrass.
- `gg-persist`— Daftar elemen pengujian untuk bertahan setelah uji coba. Perilaku default adalah tidak mempertahankan apa pun. Nilai yang diterima adalah:`aws.resources`,`installed.software`, dan`generated.files`.
- `gg-runtime`— Daftar nilai untuk mempengaruhi bagaimana tes berinteraksi dengan sumber daya pengujian. Nilai-nilai ini menggantikan`gg.persist`parameter. Jika default kosong, itu mengasumsikan semua sumber daya pengujian dikendalikan oleh kasus uji, termasuk runtime Greengrass yang diinstal. Nilai yang diterima adalah:`aws.resources`,`installed.software`, dan`generated.files`.
- `ggc-archive`— Jalur menuju komponen inti Greengrass yang diarsipkan.
- `ggc-install-root`— Direktori untuk menginstal komponen inti Greengrass. Default ke test.temp.path dan test run folder.
- `ggc-log-level`— Atur level log nukleus Greengrass untuk uji coba. Defaultnya adalah “INFO”.
- `ggc-tes-rolename`Peran IAM yangAWS IoT GreengrassCore akan berasumsi untuk mengaksesAWSlayanan. Jika peran dengan nama yang diberikan tidak ada maka satu akan dibuat dan kebijakan akses default.
- `ggc-trusted-plugins`— Daftar koma terpisah dari jalur (pada host) dari plugin tepercaya yang perlu ditambahkan ke Greengrass. Untuk menyediakan jalur pada DUT itu sendiri, awali jalur dengan 'dut: '
- `ggc-user-name`— Nilai user:group POSIXuser untuk inti Greengrass. Default ke nama pengguna saat ini yang masuk.
- `ggc-version`— Mengganti versi komponen inti Greengrass yang sedang berjalan. Default ke nilai yang ditemukan di ggc.archive.
- `log-level`— Tingkat log uji coba. Default ke “INFO”.
- `parallel-config`— Set indeks batch dan jumlah batch sebagai JSON String. Nilai default indeks batch adalah 0 dan jumlah batch adalah 1.
- `proxy-url`— Konfigurasikan semua tes untuk merutekan lalu lintas melalui URL ini.

- `tags`— Hanya jalankan tag fitur. Dapat berpotongan dengan '&'
- `test-id-prefix`— Awalan umum yang diterapkan ke semua sumber daya khusus pengujian termasukAWSnama sumber daya dan tag. Default adalah awalan “gg”.
- `test-log-path`— Direktori yang akan berisi hasil dari seluruh uji coba. Default ke “TestResults”.
- `test-results-json`— Tandai untuk menentukan apakah laporan Cucumber JSON yang dihasilkan ditulis ke disk. Default ke true.
- `test-results-log`— Tandai untuk menentukan apakah output konsol dihasilkan ditulis ke disk. Default ke false.
- `test-results-xml`- Tandai untuk menentukan apakah laporan XMLJunit yang dihasilkan dihasilkan ditulis ke disk. Default ke true.
- `test-temp-path`— Direktori untuk menghasilkan artefak uji lokal. Default ke direktori temp acak diawali dengan gg-testing.
- `timeout-multiplier`— Pengganda disediakan untuk semua batas waktu pengujian. Defaultnya adalah 1.0.

Tutorial: Jalankan end-to-end tes menggunakan Greengrass Testing Framework dan Greengrass Development Kit

AWS IoT GreengrassTesting Framework (GTF) dan Greengrass Development Kit (GDK) menawarkan pengembang cara untuk menjalankan pengujian. end-to-end Anda dapat menyelesaikan tutorial ini untuk menginisialisasi proyek GDK dengan komponen, menginisialisasi proyek GDK dengan modul end-to-end pengujian, dan membangun kasus uji khusus. Setelah Anda membangun kasus uji kustom Anda, Anda kemudian dapat menjalankan pengujian.

Dalam tutorial ini, Anda melakukan hal-hal berikut:

1. Inisialisasi proyek GDK dengan komponen.
2. Inisialisasi proyek GDK dengan modul end-to-end uji.
3. Buat kasus uji khusus.
4. Tambahkan tag ke kasus uji baru.
5. Bangun JAR tes.
6. Jalankan tes.

Topik

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut:

- GDK versi 1.3.0 atau yang lebih baru
- Java
- Maven
- Git

Langkah 1: Inisialisasi proyek GDK dengan komponen

- Inisialisasi folder kosong dengan proyek GDK. Unduh `HelloWorld` komponen yang diimplementasikan dengan Python dengan menjalankan perintah berikut.

  ```
  gdk component init -t HelloWorld -l python -n HelloWorld
  ```

  Perintah ini membuat direktori baru bernama `HelloWorld` dalam direktori saat ini.

Langkah 2: Inisialisasi proyek GDK dengan modul uji end-to-end

- GDK memungkinkan Anda mengunduh templat modul pengujian yang terdiri dari implementasi fitur dan langkah. Jalankan perintah berikut untuk membuka `HelloWorld` direktori dan menginisialisasi proyek GDK yang ada menggunakan modul pengujian.

  ```
  cd HelloWorld
  gdk test-e2e init
  ```

Perintah ini membuat direktori baru bernama `gg-e2e-tests` dalam `HelloWorld` direktori. Direktori pengujian ini adalah proyek Maven yang memiliki ketergantungan pada JAR mandiri pengujian Greengrass.

Langkah 3: Bangun kasus uji khusus

Menulis kasus uji khusus secara luas terdiri dari dua langkah: membuat file fitur dengan skenario pengujian dan menerapkan definisi langkah. Untuk contoh pembuatan kasus uji kustom, lihatContoh: Membangun kasus uji khusus. Gunakan langkah-langkah berikut untuk membuat kasus uji kustom Anda:

1. Buat file fitur dengan skenario pengujian

   Sebuah fitur biasanya menggambarkan fungsionalitas spesifik dari perangkat lunak yang sedang diuji. Di Cucumber, setiap fitur ditentukan sebagai file fitur individual dengan judul, deskripsi rinci, dan satu atau lebih contoh kasus tertentu yang disebut skenario. Setiap skenario terdiri dari judul, deskripsi rinci, dan serangkaian langkah yang menentukan interaksi dan hasil yang diharapkan. Skenario ditulis dalam format terstruktur menggunakan kata kunci “diberikan,” “kapan,” dan “lalu”.
2. Menerapkan definisi langkah

   Definisi langkah menghubungkan langkah Gherkin dalam bahasa sederhana ke kode program. Ketika Mentimun mengidentifikasi langkah Gherkin dalam skenario, itu akan mencari definisi langkah yang cocok untuk dijalankan.

Langkah 4: Tambahkan tag ke kasus uji baru

- Anda dapat menetapkan tag ke fitur dan skenario untuk mengatur proses pengujian. Anda dapat menggunakan tag untuk mengkategorikan subset skenario dan juga memilih kait secara kondisional untuk dijalankan. Fitur dan skenario dapat memiliki beberapa tag yang dipisahkan oleh spasi.

  Dalam contoh ini, kita menggunakan `HelloWorld` komponen.

  Dalam file fitur, tambahkan tag baru bernama `@HelloWorld` di samping `@Sample` tag.

  ```
  @Sample @HelloWorld
  Scenario: As a developer, I can create a component and deploy it on my device
  ```

```
....
```

Langkah 5: Bangun JAR tes

1. Membangun komponen. Anda harus membangun komponen sebelum membangun modul pengujian.

   ```
   gdk component build
   ```

2. Bangun modul pengujian menggunakan perintah berikut. Perintah ini akan membangun JAR pengujian di `greengrass-build` folder.

   ```
   gdk test-e2e build
   ```

Langkah 6: Jalankan tes

Saat Anda menjalankan kasus uji khusus, GTF mengotomatiskan siklus hidup pengujian bersama dengan mengelola sumber daya yang dibuat selama pengujian. Ini pertama-tama menyediakan perangkat yang sedang diuji (DUT) sebagai AWS IoT sesuatu dan menginstal perangkat lunak inti Greengrass di atasnya. Kemudian akan membuat komponen baru bernama `HelloWorld` menggunakan resep yang ditentukan di jalur itu. `HelloWorld`Komponen tersebut kemudian dikerahkan ke perangkat inti melalui penerapan hal Greengrass. Ini kemudian akan diverifikasi jika penerapan berhasil. Status penerapan akan berubah menjadi `COMPLETED` dalam waktu 3 menit jika penerapan berhasil.

1. Buka `gdk-config.json` file di direktori proyek untuk menargetkan tes dengan `HelloWorld` tag. Perbarui `test-e2e` kunci menggunakan perintah berikut.

   ```
   "test-e2e":{
     "gtf_options" : {
          "tags":"HelloWorld"
      }
   }
   ```

2. Sebelum menjalankan pengujian, Anda harus memberikan AWS kredensi ke perangkat host. GTF menggunakan kredensi ini untuk mengelola AWS sumber daya selama proses pengujian. Pastikan peran yang Anda berikan memiliki izin untuk mengotomatiskan operasi yang diperlukan yang disertakan dalam pengujian.

Jalankan perintah berikut untuk memberikan AWS kredensialnya.

- Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

3. Jalankan tes menggunakan perintah berikut.

```
gdk test-e2e run
```

Perintah ini mengunduh versi terbaru dari inti Greengrass di folder dan menjalankan pengujian `greengrass-build` menggunakannya. Perintah ini juga hanya menargetkan skenario dengan `HelloWorld` tag dan menghasilkan laporan untuk skenario tersebut. Anda akan melihat AWS sumber daya yang dibuat selama tes ini dibuang di akhir tes.

Contoh: Membangun kasus uji khusus

Example

Modul pengujian yang diunduh dalam proyek GDK terdiri dari fitur sampel dan file implementasi langkah.

Dalam contoh berikut, kami membuat file fitur untuk menguji fitur penyebaran hal dari perangkat lunak Greengrass. Kami menguji sebagian fungsionalitas fitur ini dengan skenario yang melakukan penerapan komponen melalui AWS Cloud Greengrass. Ini adalah serangkaian langkah yang membantu kita memahami interaksi dan hasil yang diharapkan dari kasus penggunaan ini.

1. Buat file fitur

   Arahkan ke `gg-e2e-tests/src/main/resources/greengrass/features` folder di direktori saat ini. Anda dapat menemukan sampel `component.feature` yang terlihat seperti contoh berikut.

   Dalam file fitur ini, Anda dapat menguji fitur penyebaran benda dari perangkat lunak Greengrass. Anda dapat menguji sebagian fungsionalitas fitur ini dengan skenario yang melakukan penerapan komponen melalui cloud Greengrass. Skenario ini adalah serangkaian langkah yang membantu memahami interaksi dan hasil yang diharapkan dari kasus penggunaan ini.

   ```
   Feature: Testing features of Greengrassv2 component

   Background:
       Given my device is registered as a Thing
       And my device is running Greengrass

   @Sample
   Scenario: As a developer, I can create a component and deploy it on my device
       When I create a Greengrass deployment with components
           HelloWorld | /path/to/recipe/file
       And I deploy the Greengrass deployment configuration
       Then the Greengrass deployment is COMPLETED on the device after 180 seconds
       And I call my custom step
   ```

   GTF berisi definisi langkah dari semua langkah berikut, kecuali untuk langkah bernama:`And I call my custom step`.

2. Menerapkan definisi langkah

   JAR mandiri GTF berisi definisi langkah dari semua langkah kecuali satu langkah:. `And I call my custom step` Anda dapat menerapkan langkah ini dalam modul pengujian.

   Arahkan ke kode sumber file pengujian. Anda dapat menautkan langkah kustom Anda menggunakan definisi langkah dengan menggunakan perintah berikut.

   ```
   @And("I call my custom step")
   public void customStep() {
       System.out.println("My custom step was called ");
   }
   ```

Tutorial: Gunakan tes kepercayaan diri dari rangkaian tes kepercayaan

AWS IoT GreengrassTesting Framework (GTF) dan Greengrass Development Kit (GDK) menawarkan pengembang cara untuk menjalankan pengujian. end-to-end Anda dapat menyelesaikan tutorial ini untuk menginisialisasi proyek GDK dengan komponen, menginisialisasi proyek GDK dengan modul end-to-end pengujian, dan menggunakan tes kepercayaan dari rangkaian pengujian kepercayaan. Setelah Anda membangun kasus uji kustom Anda, Anda kemudian dapat menjalankan pengujian.

Tes kepercayaan adalah tes generik yang disediakan oleh Greengrass yang memvalidasi perilaku komponen mendasar. Tes ini dapat dimodifikasi atau diperluas agar sesuai dengan kebutuhan komponen yang lebih spesifik.

Untuk tutorial ini kita akan menggunakan HelloWorld komponen. Jika Anda menggunakan komponen lain, ganti HelloWorld komponen dengan komponen Anda.

Dalam tutorial ini, Anda melakukan hal-hal berikut:

1. Inisialisasi proyek GDK dengan komponen.
2. Inisialisasi proyek GDK dengan modul end-to-end uji.
3. Gunakan tes dari rangkaian tes kepercayaan.
4. Tambahkan tag ke kasus uji baru.
5. Bangun JAR tes.
6. Jalankan tes.

Topik

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut:

- GDK versi 1.6.0 atau yang lebih baru
- Java
- Maven
- Git

Langkah 1: Inisialisasi proyek GDK dengan komponen

- Inisialisasi folder kosong dengan proyek GDK. Unduh `HelloWorld` komponen yang diimplementasikan dengan Python dengan menjalankan perintah berikut.

  ```
  gdk component init -t HelloWorld -l python -n HelloWorld
  ```

  Perintah ini membuat direktori baru bernama `HelloWorld` dalam direktori saat ini.

Langkah 2: Inisialisasi proyek GDK dengan modul uji end-to-end

- GDK memungkinkan Anda mengunduh templat modul pengujian yang terdiri dari implementasi fitur dan langkah. Jalankan perintah berikut untuk membuka `HelloWorld` direktori dan menginisialisasi proyek GDK yang ada menggunakan modul pengujian.

  ```
  cd HelloWorld
  gdk test-e2e init
  ```

  Perintah ini membuat direktori baru bernama `gg-e2e-tests` dalam `HelloWorld` direktori. Direktori pengujian ini adalah proyek Maven yang memiliki ketergantungan pada JAR mandiri pengujian Greengrass.

Langkah 3: Gunakan tes dari rangkaian pengujian kepercayaan

Menulis kasus uji kepercayaan terdiri dari menggunakan file fitur yang disediakan dan, jika perlu, memodifikasi skenario. Untuk contoh menggunakan tes kepercayaan diri, lihatContoh: Membangun kasus uji khusus. Gunakan langkah-langkah berikut untuk menggunakan tes kepercayaan diri:

- Gunakan file fitur yang disediakan.

  Arahkan ke `gg-e2e-tests/src/main/resources/greengrass/features` folder di direktori saat ini. Buka `confidenceTest.feature` file sampel untuk menggunakan tes kepercayaan.

Langkah 4: Tambahkan tag ke kasus uji baru

- Anda dapat menetapkan tag ke fitur dan skenario untuk mengatur proses pengujian. Anda dapat menggunakan tag untuk mengkategorikan subset skenario dan juga memilih kait secara kondisional untuk dijalankan. Fitur dan skenario dapat memiliki beberapa tag yang dipisahkan oleh spasi.

  Dalam contoh ini, kita menggunakan `HelloWorld` komponen.

  Setiap skenario ditandai dengan`@ConfidenceTest`. Ubah atau tambahkan tag jika Anda hanya ingin menjalankan sebagian dari rangkaian pengujian. Setiap skenario tes dijelaskan di bagian atas setiap tes kepercayaan. Skenario ini adalah serangkaian langkah yang membantu memahami interaksi dan hasil yang diharapkan dari setiap kasus uji. Anda dapat memperpanjang tes ini dengan menambahkan langkah Anda sendiri atau dengan memodifikasi yang sudah ada.

  ```
  @ConfidenceTest
  Scenario: As a Developer, I can deploy GDK_COMPONENT_NAME to my device and see it
   is working as expected
  ....
  ```

Langkah 5: Bangun JAR tes

1. Membangun komponen. Anda harus membangun komponen sebelum membangun modul pengujian.

   ```
   gdk component build
   ```

2. Bangun modul pengujian menggunakan perintah berikut. Perintah ini akan membangun JAR pengujian di `greengrass-build` folder.

   ```
   gdk test-e2e build
   ```

Langkah 6: Jalankan tes

Saat Anda menjalankan uji kepercayaan diri, GTF mengotomatiskan siklus hidup pengujian bersama dengan mengelola sumber daya yang dibuat selama pengujian. Ini pertama-tama menyediakan perangkat yang sedang diuji (DUT) sebagai AWS IoT sesuatu dan menginstal perangkat lunak inti Greengrass di atasnya. Kemudian akan membuat komponen baru bernama `HelloWorld` menggunakan resep yang ditentukan di jalur itu. `HelloWorld`Komponen tersebut kemudian dikerahkan ke perangkat inti melalui penerapan hal Greengrass. Ini kemudian akan diverifikasi jika penerapan berhasil. Status penerapan akan berubah menjadi `COMPLETED` dalam waktu 3 menit jika penerapan berhasil.

1. Pergi ke `gdk-config.json` file di direktori proyek untuk menargetkan tes dengan `ConfidenceTest` tag atau tag mana pun yo8u yang ditentukan dalam Langkah 4. Perbarui `test-e2e` kunci menggunakan perintah berikut.

   ```
   "test-e2e":{
     "gtf_options" : {
          "tags":"ConfidenceTest"
      }
   }
   ```

2. Sebelum menjalankan pengujian, Anda harus memberikan AWS kredensi ke perangkat host. GTF menggunakan kredensi ini untuk mengelola AWS sumber daya selama proses pengujian. Pastikan peran yang Anda berikan memiliki izin untuk mengotomatiskan operasi yang diperlukan yang disertakan dalam pengujian.

   Jalankan perintah berikut untuk memberikan AWS kredensialnya.

   - Linux or Unix

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

     Windows Command Prompt (CMD)

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

PowerShell

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

3. Jalankan tes menggunakan perintah berikut.

```
gdk test-e2e run
```

Perintah ini mengunduh versi terbaru dari inti Greengrass di folder dan menjalankan pengujian `greengrass-build` menggunakannya. Perintah ini juga hanya menargetkan skenario dengan `ConfidenceTest` tag dan menghasilkan laporan untuk skenario tersebut. Anda akan melihat AWS sumber daya yang dibuat selama tes ini dibuang di akhir tes.

Contoh: Gunakan tes kepercayaan

Example

Modul pengujian yang diunduh dalam proyek GDK terdiri dari file fitur yang disediakan.

Dalam contoh berikut, kami menggunakan file fitur untuk menguji fitur penyebaran benda dari perangkat lunak Greengrass. Kami menguji sebagian fungsionalitas fitur ini dengan skenario yang melakukan penerapan komponen melalui AWS Cloud Greengrass. Ini adalah serangkaian langkah yang membantu kita memahami interaksi dan hasil yang diharapkan dari kasus penggunaan ini.

- Gunakan file fitur yang disediakan.

  Arahkan ke `gg-e2e-tests/src/main/resources/greengrass/features` folder di direktori saat ini. Anda dapat menemukan sampel `confidenceTest.feature` yang terlihat seperti contoh berikut.

```
Feature: Confidence Test Suite

Background:
    Given my device is registered as a Thing
    And my device is running Greengrass

@ConfidenceTest
Scenario: As a Developer, I can deploy GDK_COMPONENT_NAME to my device and see it
 is working as expected
```

```
    When I create a Greengrass deployment with components
      | GDK_COMPONENT_NAME | GDK_COMPONENT_RECIPE_FILE |
      | aws.greengrass.Cli | LATEST                    |
    And I deploy the Greengrass deployment configuration
    Then the Greengrass deployment is COMPLETED on the device after 180 seconds
    # Update component state accordingly. Possible states: {RUNNING, FINISHED,
 BROKEN, STOPPING}
    And I verify the GDK_COMPONENT_NAME component is RUNNING using the greengrass-
cli
```

Setiap skenario tes dijelaskan di bagian atas setiap tes kepercayaan. Skenario ini adalah serangkaian langkah yang membantu memahami interaksi dan hasil yang diharapkan dari setiap kasus uji. Anda dapat memperpanjang tes ini dengan menambahkan langkah Anda sendiri atau dengan memodifikasi yang sudah ada. Masing-masing skenario mencakup komentar yang membantu Anda melakukan penyesuaian ini.

Kembangkan AWS IoT Greengrass komponen

Anda dapat mengembangkan dan menguji komponen pada perangkat inti Greengrass Anda. Akibatnya, Anda dapat membuat dan mengulangi AWS IoT Greengrass perangkat lunak Anda tanpa berinteraksi dengan perangkat lunak. AWS Cloud Setelah menyelesaikan versi komponen, Anda dapat mengunggahnya ke AWS IoT Greengrass di cloud, sehingga Anda dan tim dapat men-deploy komponen itu ke perangkat lain di armada Anda. Untuk informasi selengkapnya tentang cara men-deploy komponen, lihat Deploy komponen AWS IoT Greengrass ke perangkat.

Setiap komponen terdiri dari resep dan artifact.

- Resep

  Setiap komponen berisi file resep, yang mendefinisikan metadata nya. Resep ini juga menentukan parameter konfigurasi komponen, dependensi komponen, siklus hidup, dan kompatibilitas platform. Siklus hidup komponen menentukan perintah yang menginstal, menjalankan, dan menutup komponen. Untuk informasi selengkapnya, lihat AWS IoT Greengrass referensi resep komponen.

  Anda bisa menentukan resep dalam format JSON atau YAML.

- Artefak

Komponen dapat memiliki sejumlah artifact, yang merupakan komponen biner. Artifact dapat mencakup skrip, kode yang dikompilasi, sumber daya statis, dan file lain yang dikonsumsi komponen. Komponen juga dapat mengonsumsi artefak dari dependensi komponen.

AWS IoT Greengrass menyediakan komponen yang telah dibuat sebelumnya yang dapat Anda gunakan di aplikasi dan di-deploy ke perangkat Anda. Misalnya, Anda dapat menggunakan komponen pengelola aliran untuk mengunggah data ke berbagai AWS layanan, atau Anda dapat menggunakan komponen CloudWatch metrik untuk mempublikasikan metrik khusus ke Amazon. CloudWatch Untuk informasi selengkapnya, lihat Komponen yang disediakan oleh AWS.

AWS IoT Greengrassmengkurasi indeks komponen Greengrass, yang disebut Greengrass Software Catalog. Katalog ini melacak komponen Greengrass yang dikembangkan oleh komunitas Greengrass. Dari katalog ini, Anda dapat mengunduh, memodifikasi, dan menyebarkan komponen untuk membuat aplikasi Greengrass Anda. Untuk informasi selengkapnya, lihat Komponen komunitas.

Perangkat lunak inti AWS IoT Greengrass menjalankan komponen sebagai pengguna sistem dan grup, seperti `ggc_user` dan `ggc_group`, yang Anda konfigurasi pada perangkat inti. Ini berarti bahwa komponen memiliki izin dari pengguna sistem itu. Jika Anda menggunakan pengguna sistem tanpa direktori home, maka komponen tidak dapat menggunakan perintah atau kode run yang menggunakan direktori home. Ini berarti Anda tidak dapat menggunakan perintah `pip install some-library --user` untuk menginstal paket Python misalnya. Jika Anda mengikuti Memulai tutorial untuk mengatur perangkat inti, maka pengguna sistem Anda tidak memiliki direktori home. Untuk informasi selengkapnya tentang cara mengonfigurasi pengguna dan grup yang menjalankan komponen, lihat Konfigurasikan pengguna yang menjalankan komponen.

> **Note**
>
> AWS IoT Greengrass menggunakan versi semantik untuk komponen. Versi semantik mengikuti sistem nomor mayor.minor.patch. Sebagai contoh, versi `1.0.0` merupakan rilis mayor pertama untuk sebuah komponen. Untuk informasi lebih lanjut, lihat spesifikasi versi semantik.

Topik

- Siklus hidup komponen
- Jenis komponen

- Buat AWS IoT Greengrass komponen
- Uji AWS IoT Greengrass komponen dengan penerapan lokal
- Publikasikan komponen untuk diterapkan ke perangkat inti Anda
- Berinteraksilah dengan layanan AWS
- Jalankan kontainer Docker
- AWS IoT Greengrass referensi resep komponen
- Referensi variabel lingkungan komponen

Siklus hidup komponen

Siklus hidup komponen mendefinisikan tahapan yang digunakan oleh Perangkat lunak inti AWS IoT Greengrass untuk menginstal dan menjalankan komponen. Setiap tahap mendefinisikan skrip dan informasi lain yang menentukan bagaimana komponen berperilaku. Misalnya, saat Anda menginstal komponen, perangkat lunak inti AWS IoT Greengrass akan menjalankan skrip siklus hidup `Install` untuk komponen tersebut. Komponen pada perangkat inti memiliki status siklus hidup berikut:

- `NEW` – Resep dan artefak komponen dimuat pada perangkat inti, namun komponennya tidak terpasang. Setelah komponen memasuki status ini, ia menjalankan skrip instalasinya.
- `INSTALLED` – Komponen dipasang pada perangkat inti. Komponen memasuki keadaan ini setelah menjalankan install scripts.
- `STARTING` – Komponen mulai pada perangkat inti. Komponen memasuki keadaan ini ketika menjalankan install scripts. Jika permulaan ini berhasil, komponen akan memasuki keadaan `RUNNING`.
- `RUNNING` – Komponen mulai pada perangkat inti. Komponen memasuki keadaan ini ketika menjalankan run script atau saat memiliki proses latar belakang aktif dari startup script.
- `FINISHED`— Komponen berjalan dengan sukses dan menyelesaikan operasinya.
- `STOPPING` – Komponen berhenti. Komponen memasuki keadaan ini ketika menjalankan shutdown scripts.
- `ERRORED` — Komponen mengalami kesalahan. Ketika komponen memasuki keadaan ini, ia akan menjalankan skrip pemulihan. Kemudian, komponen restart untuk mencoba kembali ke penggunaan normal. Jika komponen memasuki keadaan `ERRORED` tiga kali tanpa berjalan sukses, komponen menjadi `BROKEN`.
- `BROKEN` — Komponen mengalami kesalahan beberapa kali dan tidak dapat pulih. Anda harus men-deploy komponen lagi untuk memperbaikinya.

Jenis komponen

Jenis komponen menentukan bagaimana Perangkat lunak inti AWS IoT Greengrass menjalankan komponen. Komponen dapat memiliki jenis berikut:

- Inti (`aws.greengrass.nucleus`)

  Inti Greengrass adalah komponen yang menyediakan fungsionalitas minimum perangkat lunak inti AWS IoT Greengrass. Untuk informasi selengkapnya, lihat Inti Greengrass.

- Plugin (`aws.greengrass.plugin`)

  Inti Greengrass menjalankan komponen plugin dalam Java Virtual Machine (JVM) yang sama sebagai inti. Nukleus dimulai ulang saat Anda mengubah versi komponen plugin pada perangkat inti. Untuk menginstal dan menjalankan komponen plugin, Anda harus mengonfigurasi inti Greengrass agar berjalan sebagai layanan sistem. Untuk informasi selengkapnya, lihat Konfigurasikan inti Greengrass sebagai layanan sistem.

  Beberapa komponen yang disediakan oleh AWS adalah komponen plugin, yang memungkinkannya untuk terhubung dengan antarmuka secara langsung dengan inti Greengrass. Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

- Generik (`aws.greengrass.generic`)

  Inti Greengrass menjalankan skrip siklus hidup komponen generik, jika komponen mendefinisikan siklus hidup.

  Jenis ini adalah jenis default untuk komponen kustom.

- Lambda () `aws.greengrass.lambda`

  Inti Greengrass menjalankan komponen fungsi Lambda menggunakan Komponen peluncur Lambda.

  Bila Anda membuat komponen dari fungsi Lambda, komponen memiliki jenis ini. Untuk informasi selengkapnya, lihat Jalankan fungsi AWS Lambda.

> **Note**
>
> Kami tidak menyarankan Anda untuk menentukan jenis komponen dalam resep. AWS IoT Greengrass menetapkan jenis itu untuk Anda ketika Anda membuat komponen.

Buat AWS IoT Greengrass komponen

Anda dapat mengembangkan AWS IoT Greengrass komponen khusus pada komputer pengembangan lokal atau perangkat inti Greengrass. AWS IoT Greengrass menyediakan AWS IoT Greengrass Development Kit Command-Line Interface (GDK CLI) untuk membantu Anda membuat, membangun, dan menerbitkan komponen dari templat komponen dan komponen komunitas yang telah ditentukan sebelumnya. Anda juga dapat menjalankan perintah shell bawaan untuk membuat, membangun, dan menerbitkan komponen. Pilih dari opsi berikut untuk membuat komponen Greengrass kustom:

- Gunakan CLI Kit Pengembangan Greengrass

  Gunakan CLI GDK untuk mengembangkan komponen pada komputer pengembangan lokal. CLI GDK membangun dan mengemas kode sumber komponen menjadi resep dan artefak yang dapat Anda publikasikan sebagai komponen pribadi ke layanan. AWS IoT Greengrass Anda dapat mengonfigurasi CLI GDK untuk memperbarui versi komponen dan URI artefak secara otomatis saat Anda mempublikasikan komponen, jadi Anda tidak perlu memperbarui resep setiap kali. Untuk mengembangkan komponen menggunakan CLI GDK, Anda dapat memulai dari template atau komponen komunitas dari Katalog Perangkat Lunak Greengrass. Untuk informasi selengkapnya, lihat AWS IoT GreengrassKit Pengembangan Antarmuka Baris Perintah.
- Jalankan perintah shell bawaan

  Anda dapat menjalankan perintah shell bawaan untuk mengembangkan komponen pada komputer pengembangan lokal atau pada perangkat inti Greengrass. Anda menggunakan perintah shell untuk menyalin atau membangun kode sumber komponen menjadi artefak. Setiap kali Anda membuat versi baru komponen, Anda harus membuat atau memperbarui resep dengan versi komponen baru. Saat memublikasikan komponen ke AWS IoT Greengrass layanan, Anda harus memperbarui URI ke setiap artefak komponen dalam resep.

Topik

- Buat komponen (GDK CLI)

- Buat komponen (perintah shell)

Buat komponen (GDK CLI)

Ikuti petunjuk di bagian ini untuk membuat dan membangun komponen menggunakan CLI GDK.

Untuk mengembangkan komponen Greengrass (GDK CLI)

1. Jika Anda belum melakukannya, instal CLI GDK di komputer pengembangan Anda. Untuk informasi selengkapnya, lihat Instal atau perbarui Antarmuka AWS IoT Greengrass Baris Perintah Kit Pengembangan.
2. Ubah ke folder tempat Anda ingin membuat folder komponen.

   Linux or Unix

   ```
   mkdir ~/greengrassv2
   cd ~/greengrassv2
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir %USERPROFILE%\greengrassv2
   cd %USERPROFILE%\greengrassv2
   ```

   PowerShell

   ```
   mkdir ~/greengrassv2
   cd ~/greengrassv2
   ```

3. Pilih template komponen atau komponen komunitas untuk diunduh. CLI GDK mengunduh template atau komponen komunitas, sehingga Anda dapat memulai dari contoh fungsional. Gunakan perintah daftar komponen untuk mengambil daftar template yang tersedia atau komponen komunitas.
   - Untuk daftar template komponen, jalankan perintah berikut. Setiap baris dalam respons mencakup nama template dan bahasa pemrograman.

     ```
     gdk component list --template
     ```

   - Untuk daftar komponen komunitas, jalankan perintah berikut.

```
gdk component list --repository
```

4. Buat dan ubah ke folder komponen tempat CLI GDK mengunduh templat atau komponen komunitas. Ganti *HelloWorld*dengan nama komponen, atau nama lain yang membantu Anda mengidentifikasi folder komponen ini.

   Linux or Unix

   ```
   mkdir HelloWorld
   cd HelloWorld
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir HelloWorld
   cd HelloWorld
   ```

   PowerShell

   ```
   mkdir HelloWorld
   cd HelloWorld
   ```

5. Unduh template atau komponen komunitas ke folder saat ini. Gunakan perintah komponen init.

   - Untuk membuat folder komponen dari template, jalankan perintah berikut. Ganti *HelloWorld*dengan nama template, dan ganti *python* dengan nama bahasa pemrograman.

     ```
     gdk component init --template HelloWorld --language python
     ```

   - Untuk membuat folder komponen dari komponen komunitas, jalankan perintah berikut. Ganti *ComponentName*dengan nama komponen komunitas.

     ```
     gdk component init --repository ComponentName
     ```

   > **Note**
   >
   > Jika Anda menggunakan GDK CLI v1.0.0, Anda harus menjalankan perintah ini di folder kosong. CLI GDK mengunduh template atau komponen komunitas ke folder saat ini.

Jika Anda menggunakan GDK CLI v1.1.0 atau yang lebih baru, Anda dapat menentukan `--name` argumen untuk menentukan folder tempat CLI GDK mengunduh template atau komponen komunitas. Jika Anda menggunakan argumen ini, tentukan folder yang tidak ada. CLI GDK membuat folder untuk Anda. Jika Anda tidak menentukan argumen ini, CLI GDK menggunakan folder saat ini, yang harus kosong.

6. CLI GDK membaca dari file konfigurasi CLI GDK, `gdk-config.json` bernama, untuk membangun dan menerbitkan komponen. File konfigurasi ini ada di root folder komponen. Langkah sebelumnya membuat file ini untuk Anda. Pada langkah ini, Anda memperbarui `gdk-config.json` dengan informasi tentang komponen Anda. Lakukan hal-hal berikut:

   a. Buka `gdk-config.json` di editor teks.

   b. (Opsional) Ubah nama komponen. Nama komponen adalah kunci dalam `component` objek.

   c. Ubah penulis komponen.

   d. (Opsional) Ubah versi komponen. Tentukan satu dari yang berikut ini:

      - `NEXT_PATCH`— Saat Anda memilih opsi ini, CLI GDK menetapkan versi saat Anda mempublikasikan komponen. CLI GDK menanyakan layanan AWS IoT Greengrass untuk mengidentifikasi versi komponen terbaru yang diterbitkan. Kemudian, ia menetapkan versi ke versi patch berikutnya setelah versi itu. Jika Anda belum mempublikasikan komponen sebelumnya, CLI GDK menggunakan versi. `1.0.0`

        Jika Anda memilih opsi ini, Anda tidak dapat menggunakan CLI Greengrass untuk menyebarkan dan menguji komponen secara lokal ke komputer pengembangan lokal Anda yang menjalankan perangkat lunak Core. AWS IoT Greengrass Untuk mengaktifkan penerapan lokal, Anda harus menentukan versi semantik sebagai gantinya.

      - Versi semantik, seperti. **`1.0.0`** Versi semantik menggunakan mayor. kecil. sistem penomoran tambalan. Untuk informasi lebih lanjut, lihat spesifikasi versi semantik.

        Jika Anda mengembangkan komponen pada perangkat inti Greengrass tempat Anda ingin menerapkan dan menguji komponen, pilih opsi ini. Anda harus membangun komponen dengan versi tertentu untuk membuat penerapan lokal dengan CLI Greengrass.

   e. (Opsional) Ubah konfigurasi build untuk komponen. Konfigurasi build mendefinisikan bagaimana CLI GDK membangun sumber komponen menjadi artefak. Pilih dari opsi berikut untuk`build_system`:

- `zip`— Paket folder komponen ke dalam file ZIP untuk didefinisikan sebagai satu-satunya artefak komponen. Pilih opsi ini untuk jenis komponen berikut:
  - Komponen yang menggunakan bahasa pemrograman yang ditafsirkan, seperti JavaScript Python atau.
  - Komponen yang mengemas file selain kode, seperti model pembelajaran mesin atau sumber daya lainnya.

  CLI GDK meritsleting folder komponen ke dalam file zip dengan nama yang sama dengan folder komponen. Misalnya, jika nama folder komponen adalah`HelloWorld`, CLI GDK membuat file zip bernama. `HelloWorld.zip`

  > **Note**
  >
  > Jika Anda menggunakan GDK CLI versi 1.0.0 pada perangkat Windows, folder komponen dan nama file zip harus berisi hanya huruf kecil.

  Ketika CLI GDK meritsleting folder komponen ke dalam file zip, ia melewatkan file-file berikut:
  - File `gdk-config.json`
  - File resep (`recipe.jsonataurecipe.yaml`)
  - Membangun folder, seperti `greengrass-build`
- `maven`— Menjalankan `mvn clean package` perintah untuk membangun sumber komponen menjadi artefak. Pilih opsi ini untuk komponen yang menggunakan Maven, seperti komponen Java.

  Pada perangkat Windows, fitur ini tersedia untuk GDK CLI v1.1.0 dan yang lebih baru.
- `gradle`— Menjalankan `gradle build` perintah untuk membangun sumber komponen menjadi artefak. Pilih opsi ini untuk komponen yang menggunakan Gradle. Fitur ini tersedia untuk GDK CLI v1.1.0 dan yang lebih baru.

  Sistem `gradle` build mendukung Kotlin DSL sebagai file build. Fitur ini tersedia untuk GDK CLI v1.2.0 dan yang lebih baru.
- `gradlew`— Menjalankan `gradlew` perintah untuk membangun sumber komponen menjadi artefak. Pilih opsi ini untuk komponen yang menggunakan Gradle Wrapper.

Fitur ini tersedia untuk GDK CLI v1.2.0 dan yang lebih baru.

- `custom`— Menjalankan perintah khusus untuk membangun sumber komponen menjadi resep dan artefak. Tentukan perintah khusus dalam `custom_build_command` parameter.

f. Jika Anda menentukan `custom` untuk`build_system`, tambahkan `custom_build_command` ke `build` objek. Dalam`custom_build_command`, tentukan satu string atau daftar string, di mana setiap string adalah kata dalam perintah. Misalnya, untuk menjalankan perintah build kustom untuk komponen C++, Anda dapat menentukan**`["cmake", "--build", "build", "--config", "Release"]`**.

g. Jika Anda menggunakan GDK CLI v1.1.0 atau yang lebih baru, Anda dapat menentukan argumen untuk menentukan `--bucket` bucket S3 tempat CLI GDK mengunggah artefak komponen. Jika Anda tidak menentukan argumen ini, CLI GDK akan mengunggah ke bucket S3 yang namanya, di *`mana`* bucket *`dan region bucket-region-accountId adalah nilai yang Anda tentukan`*`gdk-config.json`*`, dan accountID`* adalah ID Anda. Akun AWS CLI GDK membuat bucket jika tidak ada.

   Ubah konfigurasi publikasi untuk komponen. Lakukan hal-hal berikut:

   i. Tentukan nama bucket S3 yang akan digunakan untuk meng-host artefak komponen.

   ii. Tentukan Wilayah AWS tempat CLI GDK menerbitkan komponen.

Ketika Anda selesai dengan langkah ini, `gdk-config.json` file mungkin terlihat mirip dengan contoh berikut.

```
{
  "component": {
    "com.example.PythonHelloWorld": {
      "author": "Amazon",
      "version": "NEXT_PATCH",
      "build": {
        "build_system" : "zip"
      },
      "publish": {
        "bucket": "greengrass-component-artifacts",
        "region": "us-west-2"
      }
    }
  },
```

```
  "gdk_version": "1.0.0"
}
```

7. Perbarui file resep komponen, bernama `recipe.yaml` atau`recipe.json`. Lakukan hal-hal berikut:

   a. Jika Anda mengunduh templat atau komponen komunitas yang menggunakan sistem `zip` build, periksa apakah nama artefak zip cocok dengan nama folder komponen. CLI GDK meritsleting folder komponen ke dalam file zip dengan nama yang sama dengan folder komponen. Resep berisi nama artefak zip dalam daftar artefak komponen dan skrip siklus hidup yang menggunakan file dalam artefak zip. Perbarui `Artifacts` dan `Lifecycle` definisi sedemikian rupa sehingga nama file zip cocok dengan nama folder komponen. Contoh resep parsi berikut menyoroti nama file zip dalam `Lifecycle` definisi `Artifacts` dan.

      JSON

      ```
      {
        ...
        "Manifests": [
          {
            "Platform": {
              "os": "all"
            },
            "Artifacts": [
              {
                "URI": "s3://{COMPONENT_NAME}/{COMPONENT_VERSION}/HelloWorld.zip",
                "Unarchive": "ZIP"
              }
            ],
            "Lifecycle": {
              "run": "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
       {configuration:/Message}"
            }
          }
        ]
      }
      ```

      YAML

      ```
      ---
      ```

```
...
Manifests:
  - Platform:
      os: all
    Artifacts:
      - URI: "s3://BUCKET_NAME/COMPONENT_NAME/
COMPONENT_VERSION/HelloWorld.zip"
        Unarchive: ZIP
    Lifecycle:
      run: "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
 {configuration:/Message}"
```

b. (Opsional) Perbarui deskripsi komponen, konfigurasi default, artefak, skrip siklus hidup, dan dukungan platform. Untuk informasi selengkapnya, lihat AWS IoT Greengrass referensi resep komponen.

Setelah selesai dengan langkah ini, file resep mungkin terlihat mirip dengan contoh berikut.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "{COMPONENT_NAME}",
  "ComponentVersion": "{COMPONENT_VERSION}",
  "ComponentDescription": "This is a simple Hello World component written in
 Python.",
  "ComponentPublisher": "{COMPONENT_AUTHOR}",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "Message": "World"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "all"
      },
      "Artifacts": [
        {
          "URI": "s3://{COMPONENT_NAME}/{COMPONENT_VERSION}/HelloWorld.zip",
          "Unarchive": "ZIP"
        }
```

```
      ],
      "Lifecycle": {
        "run": "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
 {configuration:/Message}"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: "2020-01-25"
ComponentName: "{COMPONENT_NAME}"
ComponentVersion: "{COMPONENT_VERSION}"
ComponentDescription: "This is a simple Hello World component written in
 Python."
ComponentPublisher: "{COMPONENT_AUTHOR}"
ComponentConfiguration:
  DefaultConfiguration:
    Message: "World"
Manifests:
  - Platform:
      os: all
    Artifacts:
      - URI: "s3://BUCKET_NAME/COMPONENT_NAME/COMPONENT_VERSION/HelloWorld.zip"
        Unarchive: ZIP
    Lifecycle:
      run: "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
 {configuration:/Message}"
```

8. Kembangkan dan bangun komponen Greengrass. Perintah build komponen menghasilkan resep dan artefak di `greengrass-build` folder di folder komponen. Jalankan perintah berikut.

```
gdk component build
```

Saat Anda siap untuk menguji komponen Anda, gunakan CLI GDK untuk mempublikasikannya ke AWS IoT Greengrass layanan. Kemudian, Anda dapat menerapkan komponen ke perangkat inti Greengrass. Untuk informasi selengkapnya, lihat Publikasikan komponen untuk diterapkan ke perangkat inti Anda.

Buat komponen (perintah shell)

Ikuti petunjuk di bagian ini untuk membuat folder resep dan artefak yang berisi kode sumber dan artefak untuk beberapa komponen.

Untuk mengembangkan komponen Greengrass (perintah shell)

1. Buat folder untuk komponen Anda dengan subfolder untuk resep dan artefak. Jalankan perintah berikut pada perangkat inti Greengrass Anda untuk membuat folder ini dan mengubah ke folder komponen. Ganti *~/greengrassv2* atau *%USERPROFILE%\ greengrassv2* dengan path ke folder yang akan digunakan untuk pengembangan lokal.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

2. Gunakan editor teks untuk membuat file resep yang mendefinisikan metadata, parameter, dependensi, siklus hidup, dan kemampuan platform komponen Anda. Sertakan versi komponen dalam nama file resep agar Anda dapat mengidentifikasi resep yang mencerminkan versi komponen. Anda dapat memilih format YAML atau JSON untuk resep Anda.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuat file.

   JSON

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

YAML

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

> **Note**
>
> AWS IoT Greengrass menggunakan versi semantik untuk komponen. Versi semantik mengikuti sistem nomor mayor.minor.patch. Sebagai contoh, versi `1.0.0` merupakan rilis mayor pertama untuk sebuah komponen. Untuk informasi lebih lanjut, lihat spesifikasi versi semantik.

3. Tentukan resep untuk komponen Anda. Untuk informasi selengkapnya, lihat AWS IoT Greengrass referensi resep komponen.

   Resep Anda mungkin terlihat serupa dengan contoh resep Hello World berikut.

   JSON

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

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

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.HelloWorld
ComponentVersion: '1.0.0'
ComponentDescription: My first AWS IoT Greengrass component.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    Message: world
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      run: |
        python3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
  - Platform:
      os: windows
    Lifecycle:
      run: |
        py -3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
```

Resep ini menjalankan skrip Hello World Python, yang mungkin terlihat mirip dengan contoh skrip berikut.

```
import sys

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

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

4. Buat folder untuk versi komponen yang akan dikembangkan. Kami merekomendasikan agar Anda menggunakan folder terpisah untuk setiap versi komponen artefak agar Anda dapat mengidentifikasi artefak untuk setiap versi komponen. Jalankan perintah berikut.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

   > **Important**
   >
   > Anda harus menggunakan format berikut untuk jalur folder artefak. Sertakan nama komponen dan versi yang Anda tentukan dalam resep.
   >
   > ```
   > artifacts/componentName/componentVersion/
   > ```

5. Buat artefak untuk komponen Anda di folder yang Anda buat di langkah sebelumnya. Artefak dapat mencakup perangkat lunak, gambar, dan biner lain yang menggunakan komponen Anda.

   Ketika komponen Anda sudah siap, uji komponen Anda.

Uji AWS IoT Greengrass komponen dengan penerapan lokal

Jika Anda mengembangkan komponen Greengrass pada perangkat inti, Anda dapat membuat penerapan lokal untuk menginstal dan mengujinya. Ikuti langkah-langkah di bagian ini untuk membuat penyebaran lokal.

Jika Anda mengembangkan komponen pada komputer yang berbeda, seperti komputer pengembangan lokal, Anda tidak dapat membuat penyebaran lokal. Sebagai gantinya, publikasikan komponen ke AWS IoT Greengrass layanan sehingga Anda dapat menerapkannya ke perangkat inti Greengrass untuk mengujinya. Lihat informasi yang lebih lengkap di Publikasikan komponen untuk diterapkan ke perangkat inti Anda dan Deploy komponen AWS IoT Greengrass ke perangkat.

Untuk menguji komponen pada perangkat inti Greengrass

1. Perangkat inti mencatat peristiwa seperti pembaruan komponen. Anda dapat melihat file log ini untuk menemukan dan memecahkan masalah kesalahan dengan komponen Anda, seperti resep yang tidak valid. Berkas log ini juga menampilkan pesan bahwa komponen Anda mencetak standard out (stdout). Kami sarankan Anda membuka sesi terminal tambahan pada perangkat inti Anda untuk mengamati pesan log baru secara real time. Buka sesi terminal baru, seperti melalui SSH, dan jalankan perintah berikut untuk melihat log. Ganti */greengrass/v2* dengan jalur ke folder AWS IoT Greengrass root.

   Linux or Unix

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

   PowerShell

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

   Anda juga dapat melihat file log untuk komponen Anda.

   Linux or Unix

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

   PowerShell

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

2. Dalam sesi terminal asli Anda, jalankan perintah berikut untuk memperbarui perangkat inti dengan komponen Anda. Ganti */greengrass/v2* dengan path ke folder AWS IoT Greengrass root, dan ganti *~/greengrassv2* dengan path ke folder pengembangan lokal Anda.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

> **Note**
>
> Anda juga dapat menggunakan perintah `greengrass-cli deployment create` untuk menetapkan nilai parameter konfigurasi komponen Anda. Untuk informasi selengkapnya, lihat buat.

3. Gunakan `greengrass-cli deployment status` perintah untuk memantau kemajuan penerapan komponen Anda.

   Unix or Linux

   ```
   sudo /greengrass/v2/bin/greengrass-cli deployment status \
     -i deployment-id
   ```

   Windows Command Prompt (CMD)

   ```
   C:\greengrass\v2\bin\greengrass-cli deployment status ^
   ```

```
  -i deployment-id
```

PowerShell

```
C:\greengrass\v2\bin\greengrass-cli deployment status `
  -i deployment-id
```

4. Uji komponen Anda begitu ia berjalan pada perangkat inti Greengrass. Setelah menyelesaikan versi komponen ini, Anda dapat mengunggahnya ke layanan AWS IoT Greengrass. Kemudian, Anda dapat men-deploy komponen ke perangkat inti lainnya. Lihat informasi yang lebih lengkap di Publikasikan komponen untuk diterapkan ke perangkat inti Anda.

Publikasikan komponen untuk diterapkan ke perangkat inti Anda

Setelah Anda membangun atau menyelesaikan versi komponen, Anda dapat mempublikasikannya ke AWS IoT Greengrass layanan. Kemudian, Anda dapat menerapkannya ke perangkat inti Greengrass.

Jika Anda menggunakan Greengrass Development Kit CLI (GDK CLI) untuk mengembangkan dan membangun komponen, Anda dapat menggunakan CLI GDK untuk mempublikasikan komponen ke. AWS Cloud Jika tidak, gunakan perintah shell bawaan dan AWS CLI untuk mempublikasikan komponen.

Anda juga dapat menggunakan AWS CloudFormation untuk membuat komponen dan AWS sumber daya lainnya dari template. Untuk informasi lebih lanjut, lihat Apa itu AWS CloudFormation? dan AWS::GreengrassV2::ComponentVersiondalam Panduan AWS CloudFormation Pengguna.

Topik

- Publikasikan komponen (GDK CLI)
- Publikasikan komponen (perintah shell)

Publikasikan komponen (GDK CLI)

Ikuti petunjuk di bagian ini untuk mempublikasikan komponen menggunakan CLI GDK. CLI GDK mengunggah artefak build ke bucket S3, memperbarui URI artefak dalam resep, dan membuat komponen dari resep. Anda menentukan bucket S3 dan Region yang akan digunakan dalam file konfigurasi CLI GDK.

Jika Anda menggunakan GDK CLI v1.1.0 atau yang lebih baru, Anda dapat menentukan argumen untuk menentukan `--bucket` bucket S3 tempat CLI GDK mengunggah artefak komponen. Jika Anda tidak menentukan argumen ini, CLI GDK akan mengunggah ke bucket S3 yang namanya, di *mana* bucket *dan region bucket-region-accountId adalah nilai yang Anda tentukangdk-config.json, dan accountID* adalah ID Anda. Akun AWS CLI GDK membuat bucket jika tidak ada.

> **Important**
>
> Peran perangkat inti tidak mengizinkan akses ke bucket S3 secara default. Jika ini pertama kalinya Anda menggunakan bucket S3, Anda harus menambahkan izin ke peran untuk mengizinkan perangkat inti mengambil artefak komponen dari bucket S3 ini. Untuk informasi selengkapnya, lihat Izinkan akses ke bucket S3 untuk artefak komponen.

Untuk mempublikasikan komponen Greengrass (GDK CLI)

1. Buka folder komponen di prompt perintah atau terminal.
2. Jika Anda belum melakukannya, buat komponen Greengrass. Perintah build komponen menghasilkan resep dan artefak di `greengrass-build` folder di folder komponen. Jalankan perintah berikut.

   ```
   gdk component build
   ```

3. Publikasikan komponen ke AWS Cloud. Perintah component publish mengunggah artefak komponen ke Amazon S3 dan memperbarui resep komponen dengan URI masing-masing artefak. Kemudian, itu menciptakan komponen dalam AWS IoT Greengrass layanan.

   > **Note**
   >
   > AWS IoT Greengrass menghitung intisari setiap artefak saat Anda membuat komponen. Ini berarti bahwa Anda tidak dapat memodifikasi file artefak dalam bucket S3 Anda setelah Anda membuat komponen. Jika Anda melakukannya, deployment yang mencakup komponen ini akan gagal, karena file digest tidak cocok. Jika Anda mengubah file artefak, Anda harus membuat versi baru komponen.

Jika Anda menentukan `NEXT_PATCH` versi komponen dalam file konfigurasi CLI GDK, CLI GDK menggunakan versi tambalan berikutnya yang belum ada di layanan. AWS IoT Greengrass

Jalankan perintah berikut.

```
gdk component publish
```

Output memberi tahu Anda versi komponen yang dibuat CLI GDK.

Setelah mempublikasikan komponen, Anda dapat menerapkan komponen ke perangkat inti. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat.

Publikasikan komponen (perintah shell)

Gunakan prosedur berikut untuk mempublikasikan komponen menggunakan perintah shell dan AWS Command Line Interface (AWS CLI). Ketika Anda mempublikasikan komponen, Anda melakukan hal berikut:

1. Publikasikan artefak komponen ke bucket S3.
2. Tambahkan setiap URI Amazon S3 artefak ke resep komponen.
3. Buat versi komponen AWS IoT Greengrass dari resep komponen.

> **Note**
>
> Setiap versi komponen yang Anda upload harus unik. Pastikan Anda mengunggah versi komponen yang benar, karena Anda tidak dapat mengeditnya setelah mengunggahnya.

Anda dapat mengikuti langkah-langkah ini untuk mempublikasikan komponen dari komputer pengembangan Anda atau perangkat inti Greengrass Anda.

Untuk mempublikasikan komponen (perintah shell)

1. Jika komponen menggunakan versi yang ada di AWS IoT Greengrass layanan, maka Anda harus mengubah versi komponen. Buka resep dalam editor teks, naikkan versi, dan simpan file. Pilih versi baru yang mencerminkan perubahan yang Anda buat ke komponen.

> **Note**
>
> AWS IoT Greengrass menggunakan versi semantik untuk komponen. Versi semantik mengikuti sistem nomor mayor.minor.patch. Sebagai contoh, versi `1.0.0` merupakan rilis mayor pertama untuk sebuah komponen. Untuk informasi lebih lanjut, lihat spesifikasi versi semantik.

2. Jika komponen Anda memiliki artefak, lakukan hal berikut:

   a. Publikasikan artefak komponen ke bucket S3 di file Anda. Akun AWS

   > **Tip**
   >
   > Kami merekomendasikan agar Anda menyertakan nama komponen dan versi di jalur artefak di bucket S3. Skema penamaan ini dapat membantu Anda mempertahankan artefak yang digunakan oleh versi komponen sebelumnya, sehingga Anda dapat terus mendukung versi komponen sebelumnya.

   Jalankan perintah berikut untuk mempublikasikan file artefak ke bucket S3. *Ganti DOC-EXAMPLE-BUCKET dengan nama bucket, dan ganti artifacts/com.example.HelloWorld/1.0.0/artifact.py* dengan jalur ke file artefak.

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

   > **Important**
   >
   > Peran perangkat inti tidak mengizinkan akses ke bucket S3 secara default. Jika ini pertama kalinya Anda menggunakan bucket S3, Anda harus menambahkan izin ke peran untuk mengizinkan perangkat inti mengambil artefak komponen dari bucket S3 ini. Untuk informasi selengkapnya, lihat Izinkan akses ke bucket S3 untuk artefak komponen.

   b. Tambahkan daftar bernama `Artifacts` ke resep komponen jika tidak ada. Daftar `Artifacts` muncul di setiap manifes, yang mendefinisikan persyaratan komponen pada

setiap platform yang didukungnya (atau persyaratan default komponen untuk semua platform).

c. Tambahkan setiap artefak ke daftar artefak, atau perbarui URI artefak yang ada. URI Amazon S3 terdiri atas nama bucket dan path ke objek artefak dalam bucket. URI Amazon S3 artefak Anda akan terlihat serupa dengan contoh berikut.

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

Setelah Anda menyelesaikan langkah ini, resep Anda harus memiliki daftar `Artifacts` yang terlihat seperti berikut ini.

JSON

```
{
  ...
  "Manifests": [
    {
      "Lifecycle": {
        ...
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/MyGreengrassComponent/1.0.0/
artifact.py",
          "Unarchive": "NONE"
        }
      ]
    }
  ]
}
```

> **Note**
>
> Anda dapat menambahkan `"Unarchive": "ZIP"` opsi untuk artefak ZIP untuk mengonfigurasi perangkat lunak AWS IoT Greengrass Core untuk membuka zip artefak saat komponen digunakan.

YAML

```
...
Manifests:
  - Lifecycle:
      ...
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/MyGreengrassComponent/1.0.0/
artifact.py
        Unarchive: NONE
```

> **Note**
>
> Anda dapat menggunakan `Unarchive: ZIP` opsi untuk mengonfigurasi perangkat lunak AWS IoT Greengrass Core untuk membuka zip artefak ZIP saat komponen digunakan. Untuk informasi selengkapnya tentang cara menggunakan artefak ZIP dalam komponen, lihat variabel resep Artefacts:decompressedPath.

Untuk informasi lebih lanjut tentang resep, lihat AWS IoT Greengrass referensi resep komponen.

3. Gunakan AWS IoT Greengrass konsol untuk membuat komponen dari file resep.

   Jalankan perintah berikut untuk membuat komponen dari file resep. Perintah ini membuat komponen dan menerbitkannya sebagai AWS IoT Greengrass komponen pribadi di Anda Akun AWS. Ganti *path/to/recipeFile* dengan jalur menuju file resep.

   ```
   aws greengrassv2 create-component-version --inline-recipe fileb://path/to/
   recipeFile
   ```

   Salin `arn` dari respons untuk memeriksa keadaan komponen pada langkah berikutnya.

   > **Note**
   >
   > AWS IoT Greengrass menghitung intisari setiap artefak saat Anda membuat komponen. Ini berarti bahwa Anda tidak dapat memodifikasi file artefak dalam bucket S3 Anda setelah Anda membuat komponen. Jika Anda melakukannya, deployment yang

mencakup komponen ini akan gagal, karena file digest tidak cocok. Jika Anda mengubah file artefak, Anda harus membuat versi baru komponen.

4. Setiap komponen dalam AWS IoT Greengrass layanan memiliki status. Jalankan perintah berikut untuk mengonfirmasi status versi komponen yang Anda terbitkan dalam prosedur ini. Ganti *com.example. HelloWorld*dan *1.0.0* dengan versi komponen untuk kueri. Ganti `arn` dengan ARN dari langkah sebelumnya.

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

Operasi ini mengembalikan respons yang berisi metadata komponen. Metadata berisi objek `status` yang berisi keadaan komponen dan kesalahan apa pun, jika berlaku.

Ketika keadaan komponen adalah `DEPLOYABLE`, Anda dapat men-deploy komponen ke perangkat. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat.

Berinteraksilah dengan layanan AWS

Perangkat inti Greengrass menggunakan sertifikat X.509 untuk terhubung ke AWS IoT Core menggunakan protokol autentikasi bersama TLS. Sertifikat ini memungkinkan perangkat berinteraksi dengan AWS IoT tanpa kredensial AWS, yang biasanya terdiri atas access key ID dan secret access key. Layanan AWS lainnya membutuhkan kredensial AWS dan bukan sertifikat X.509 untuk memanggil operasi API pada titik akhir layanan. AWS IoT Corememiliki penyedia kredensial yang memungkinkan perangkat untuk menggunakan sertifikat X.509 untuk mengautentikasi permintaan AWS. Penyedia kredensial AWS IoT mengautentikasi perangkat menggunakan sertifikat X.509 dan mengeluarkan kredensial AWS dalam bentuk token keamanan sementara dengan hak istimewa terbatas. Perangkat dapat menggunakan token ini untuk menandatangani dan mengautentikasi permintaan AWS. Hal ini menghilangkan kebutuhan untuk menyimpan kredensial AWS.pada perangkat inti Greengrass. Untuk informasi selengkapnya, lihat Mengotorisasi panggilan langsung ke layanan AWS di Panduan Developer AWS IoT Core.

Untuk mengambil kredensial dari AWS IoT, Greengrass, perangkat inti menggunakan peran alias AWS IoT yang menunjuk ke IAM role. IAM role ini disebut peran pertukaran token. Anda membuat peran alias dan token pertukaran peran ketika Anda menginstal perangkat lunak inti AWS IoT

Greengrass. Untuk menentukan alias peran yang digunakan perangkat inti, konfigurasikan parameter `iotRoleAlias` pada Inti Greengrass.

Penyedia kredensial AWS IoT meneruskan peran pertukaran token atas nama Anda untuk memberikan kredensial AWS ke perangkat inti. Anda dapat melampirkan kebijakan IAM yang sesuai dengan peran ini untuk memungkinkan perangkat inti Anda mengakses sumber daya AWS Anda, seperti artefak komponen dalam bucket S3. Untuk informasi lebih lanjut tentang cara mengonfigurasi peran pertukaran token, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Perangkat inti Greengrass menyimpan kredensial AWS dalam memori, dan kredensialnya kedaluwarsa setelah satu jam secara default. Jika Perangkat lunak inti AWS IoT Greengrass memulai ulang, ia harus mengambil kredensial lagi. Anda dapat menggunakan UpdateRoleAliasoperasi untuk mengonfigurasi durasi kredensialnya valid.

AWS IoT Greengrass menyediakan komponen publik, komponen layanan pertukaran token, yang dapat Anda tentukan sebagai dependensi dalam komponen kustom Anda untuk berinteraksi dengan layanan AWS. Layanan pertukaran token memberi komponen Anda variabel lingkungan, `AWS_CONTAINER_CREDENTIALS_FULL_URI`, yang menentukan URI ke server lokal yang menyediakan kredensial AWS. Saat Anda membuat klien SDK AWS, klien tersebut akan memeriksa variabel lingkungan ini dan terhubung ke server lokal untuk mengambil AWS dan menggunakannya untuk menandatangani permintaan API. Hal ini memungkinkan Anda menggunakan SDK AWS dan alat lain untuk memanggil layanan AWS dalam komponen Anda. Untuk informasi selengkapnya, lihat Layanan pertukaran token.

> **Important**
>
> Dukungan untuk memperoleh kredensial AWS dengan cara ini telah ditambahkan ke SDK AWS pada 13 Juli 2016. Komponen Anda harus menggunakan Versi SDK AWS yang dibuat pada atau setelah tanggal tersebut. Untuk informasi lebih lanjut, lihat Menggunakan SDK AWS yang didukung pada Panduan Developer Layanan Amazon Elastic Container.

Untuk memperoleh kredensial AWS dalam komponen kustom Anda, tentukan `aws.greengrass.TokenExchangeService` sebagai dependensi dalam resep komponen. Contoh resep berikut menentukan komponen yang menginstal boto3 dan menjalankan skrip Python yang menggunakan kredensial AWS dari layanan pertukaran token untuk mendaftar bucket Amazon S3.

> **Note**
>
> Untuk menjalankan komponen contoh ini, perangkat Anda harus memiliki izin `s3:ListAllMyBuckets`. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.ListS3Buckets",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that uses the token exchange service to list
 S3 buckets.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.TokenExchangeService": {
      "VersionRequirement": "^2.0.0",
      "DependencyType": "HARD"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "pip3 install --user boto3",
        "run": "python3 -u {artifacts:path}/list_s3_buckets.py"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "pip3 install --user boto3",
        "run": "py -3 -u {artifacts:path}/list_s3_buckets.py"
      }
    }
  ]
```

```
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.ListS3Buckets
ComponentVersion: '1.0.0'
ComponentDescription: A component that uses the token exchange service to list S3
 buckets.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.TokenExchangeService:
    VersionRequirement: '^2.0.0'
    DependencyType: HARD
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install:
        pip3 install --user boto3
      run: |-
        python3 -u {artifacts:path}/list_s3_buckets.py
  - Platform:
      os: windows
    Lifecycle:
      install:
        pip3 install --user boto3
      run: |-
        py -3 -u {artifacts:path}/list_s3_buckets.py
```

Komponen contoh ini menjalankan skrip Python berikut, `list_s3_buckets.py` yang berisi daftar bucket Amazon S3.

```
import boto3
import os

try:
    print("Creating boto3 S3 client...")
    s3 = boto3.client('s3')
    print("Successfully created boto3 S3 client")
except Exception as e:
```

```
    print("Failed to create boto3 s3 client. Error: " + str(e))
    exit(1)

try:
    print("Listing S3 buckets...")
    response = s3.list_buckets()
    for bucket in response['Buckets']:
        print(f'\t{bucket["Name"]}')
    print("Successfully listed S3 buckets")
except Exception as e:
    print("Failed to list S3 buckets. Error: " + str(e))
    exit(1)
```

Jalankan kontainer Docker

Anda dapat mengonfigurasi AWS IoT Greengrass komponen untuk menjalankan wadah Docker dari gambar yang disimpan di lokasi berikut:

- Repositori gambar publik dan pribadi di Amazon Elastic Container Registry (Amazon ECR)
- Repositori Hub Docker publik
- Registri Terpercaya Docker publik
- Bucket S3

Dalam komponen kustom Anda, sertakan URI gambar Docker sebagai artefak untuk mengambil gambar dan menjalankannya pada perangkat inti. Untuk gambar Amazon ECR dan Hub Docker, Anda dapat menggunakan komponen Manajer aplikasi Docker untuk men-download gambar dan mengelola kredensial untuk repositori Amazon ECR privat.

Topik

Persyaratan

Untuk menjalankan kontainer Docker dalam komponen, Anda memerlukan hal berikut:

- Sebuah perangkat inti Greengrass. Jika Anda tidak memilikinya, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2.
- Docker Engine 1.9.1 atau yang lebih baru diinstal pada perangkat inti Greengrass. Versi 20.10 adalah versi terbaru yang diverifikasi untuk bekerja dengan perangkat lunak AWS IoT Greengrass Core. Anda harus menginstal Docker langsung pada perangkat inti sebelum Anda menyebarkan komponen yang menjalankan kontainer Docker.

  > **Tip**
  >
  > Anda juga dapat mengonfigurasi perangkat inti untuk menginstal Docker Engine saat komponen diinstal. Sebagai contoh, script install berikut menginstal Docker Engine sebelum memuat gambar Docker. Skrip install ini bekerja pada distribusi Linux berbasis Debian, seperti Ubuntu. Jika Anda mengonfigurasi komponen untuk menginstal Docker Engine dengan perintah ini, Anda mungkin perlu mengatur `RequiresPrivilege` ke `true` dalam skrip siklus hidup untuk menjalankan instalasi dan perintah. `docker` Untuk informasi selengkapnya, lihat AWS IoT Greengrass referensi resep komponen.
  >
  > ```
  > apt-get install docker-ce docker-ce-cli containerd.io && docker load -i
  >  {artifacts:path}/hello-world.tar
  > ```

- Pengguna sistem yang menjalankan komponen kontainer Docker harus memiliki izin root atau administrator, atau Anda harus mengonfigurasi Docker untuk menjalankannya sebagai pengguna non-root atau non-admistrator.
  - Pada perangkat Linux, Anda dapat menambahkan pengguna ke `docker` grup untuk memanggil `docker` perintah tanpa`sudo`.
  - Pada perangkat Windows, Anda dapat menambahkan pengguna ke `docker-users` grup untuk memanggil `docker` perintah tanpa hak istimewa adminstrator.

  Linux or Unix

  Untuk menambahkan`ggc_user`, atau pengguna non-root yang Anda gunakan untuk menjalankan komponen kontainer Docker, ke `docker` grup, jalankan perintah berikut.

  ```
  sudo usermod -aG docker ggc_user
  ```

Untuk informasi selengkapnya, lihat Mengelola Docker sebagai pengguna non-root.

Windows Command Prompt (CMD)

Untuk menambahkan`ggc_user`, atau pengguna yang Anda gunakan untuk menjalankan komponen kontainer Docker, ke `docker-users` grup, jalankan perintah berikut sebagai administrator.

```
net localgroup docker-users ggc_user /add
```

Windows PowerShell

Untuk menambahkan`ggc_user`, atau pengguna yang Anda gunakan untuk menjalankan komponen kontainer Docker, ke `docker-users` grup, jalankan perintah berikut sebagai administrator.

```
Add-LocalGroupMember -Group docker-users -Member ggc_user
```

- File diakses oleh komponen kontainer Docker yang dipasang sebagai volume dalam kontainer Docker.
- Jika Anda mengkonfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan proxy jaringan, Anda harus mengkonfigurasi Docker untuk menggunakan server proxy yang sama.

Selain persyaratan ini, Anda juga harus memenuhi persyaratan berikut jika hal itu berlaku pada lingkungan Anda:

- Untuk menggunakan Docker Compose untuk membuat dan memulai kontainer Docker Anda, instal Docker Compose pada perangkat inti Greengrass Anda, dan upload file Docker Compose ke bucket S3. Anda harus menyimpan file Compose Anda dalam bucket S3 yang sama Akun AWS dan Wilayah AWS sebagai komponen. Untuk contoh yang menggunakan perintah `docker-compose up` dalam komponen kustom, lihat Jalankan kontainer Docker dari gambar publik di Amazon ECR atau Docker Hub.
- Jika Anda menjalankan AWS IoT Greengrass di belakang proxy jaringan, konfigurasikan daemon Docker untuk menggunakan server proxy.
- Jika gambar Docker Anda disimpan di Amazon ECR atau Docker Hub, sertakan komponen manajer komponen Docker sebagai dependensi dalam komponen kontainer Docker Anda. Anda harus memulai daemon Docker pada perangkat inti sebelum Anda men-deploy komponen Anda.

Selain itu, sertakan URI gambar sebagai artefak komponen. URI gambar harus dalam format `docker:registry/image[:tag|@digest]` seperti yang ditunjukkan dalam contoh berikut:

- Gambar Amazon ECR privat: `docker:account-id.dkr.ecr.region.amazonaws.com/repository/image[:tag|@digest]`
- Citra Amazon ECR publik: `docker:public.ecr.aws/repository/image[:tag|@digest]`
- Citra Hub Docker publik: `docker:name[:tag|@digest]`

Untuk informasi selengkapnya tentang menjalankan kontainer Docker dari gambar yang disimpan di repositori publik, lihat. Jalankan kontainer Docker dari gambar publik di Amazon ECR atau Docker Hub

- Jika gambar Docker Anda disimpan dalam repositori privat Amazon ECR, maka Anda harus menyertakan komponen layanan pertukaran token sebagai dependensi dalam komponen kontainer Docker. Juga, Peran perangkat Greengrass harus mengizinkan tindakan `ecr:GetAuthorizationToken`, `ecr:BatchGetImage`, dan `ecr:GetDownloadUrlForLayer`, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Untuk informasi tentang cara menjalankan kontainer Docker dari gambar yang disimpan dalam repositori privat Amazon ECR, lihat Jalankan kontainer Docker dari gambar privat di Amazon ECR.

- Untuk menggunakan gambar Docker yang disimpan dalam repositori pribadi Amazon ECR, repositori pribadi harus sama dengan perangkat inti. Wilayah AWS

- Jika citra Docker atau file Compose disimpan dalam bucket S3, peran perangkat Greengrass harus memungkinkan izin `s3:GetObject` untuk memungkinkan perangkat inti mengunduh citra tersebut sebagai artefak komponen, seperti yang ditunjukkan dalam kebijakan IAM contoh berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

  Untuk informasi tentang cara menjalankan kontainer Docker dari gambar yang disimpan dalam Amazon S3, lihat Jalankan kontainer Docker dari gambar di Amazon S3.

- Untuk menggunakan interprocess communication (IPC), AWS credentials, atau stream manager di komponen container Docker Anda, Anda harus menentukan opsi tambahan saat menjalankan container Docker. Untuk informasi selengkapnya, lihat berikut ini:
  - Gunakan interprocess communication dalam komponen kontainer Docker
  - Gunakan AWS kredensil dalam komponen wadah Docker (Linux)
  - Gunakan pengelola aliran di komponen wadah Docker (Linux)

Jalankan kontainer Docker dari gambar publik di Amazon ECR atau Docker Hub

Bagian ini menjelaskan bagaimana Anda dapat membuat komponen kustom yang menggunakan Docker Compose untuk menjalankan kontainer Docker dari gambar Docker yang disimpan di Amazon ECR dan Docker Hub.

Untuk menjalankan kontainer Docker menggunakan Docker Compose

1. Buat dan unggah file Docker Compose ke bucket Amazon S3. Pastikan bahwa peran perangkat Greengrass memungkinkan izin `s3:GetObject` untuk mengaktifkan perangkat

untuk mengakses file Compose. Contoh file Compose yang ditampilkan dalam contoh berikut mencakup image Amazon CloudWatch Agent dari Amazon ECR dan image MySQL dari Docker Hub.

```
version: "3"
services:
  cloudwatchagent:
    image: "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest"
  mysql:
    image: "mysql:8.0"
```

2. Buat komponen khusus di perangkat AWS IoT Greengrass inti Anda. Contoh resep yang ditunjukkan dalam contoh berikut memiliki sifat berikut:

   - Komponen manajer aplikasi Docker sebagai dependensi. Komponen ini memungkinkan AWS IoT Greengrass untuk mengunduh gambar dari repositori Amazon ECR dan Docker Hub publik.
   - Artefak komponen yang menentukan gambar Docker dalam repositori Amazon ECR publik.
   - Artefak komponen yang menentukan gambar Docker dalam repositori Docker Hub publik.
   - Komponen artefak yang menentukan file Docker Compose yang mencakup kontainer untuk gambar Docker yang ingin Anda jalankan.
   - Skrip lifecycle run yang menggunakan docker-compose up untuk membuat dan memulai sebuah kontainer dari citra yang ditentukan.

   JSON

   ```
   {
     "RecipeFormatVersion": "2020-01-25",
     "ComponentName": "com.example.MyDockerComposeComponent",
     "ComponentVersion": "1.0.0",
     "ComponentDescription": "A component that uses Docker Compose to run images
    from public Amazon ECR and Docker Hub.",
     "ComponentPublisher": "Amazon",
     "ComponentDependencies": {
       "aws.greengrass.DockerApplicationManager": {
         "VersionRequirement": "~2.0.0"
       }
     },
     "Manifests": [
   ```

```
    {
      "Platform": {
        "os": "all"
      },
      "Lifecycle": {
        "run": "docker-compose -f {artifacts:path}/docker-compose.yaml up"
      },
      "Artifacts": [
        {
          "URI": "docker:public.ecr.aws/cloudwatch-agent/cloudwatch-
agent:latest"
        },
        {
          "URI": "docker:mysql:8.0"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/folder/docker-compose.yaml"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.MyDockerComposeComponent
ComponentVersion: '1.0.0'
ComponentDescription: 'A component that uses Docker Compose to run images from
 public Amazon ECR and Docker Hub.'
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.DockerApplicationManager:
    VersionRequirement: ~2.0.0
Manifests:
  - Platform:
      os: all
    Lifecycle:
        run: docker-compose -f {artifacts:path}/docker-compose.yaml up
    Artifacts:
      - URI: "docker:public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest"
      - URI: "docker:mysql:8.0"
```

```
      - URI: "s3://DOC-EXAMPLE-BUCKET/folder/docker-compose.yaml"
```

> **Note**
>
> Untuk menggunakan interprocess communication (IPC), AWS credentials, atau stream manager di komponen container Docker Anda, Anda harus menentukan opsi tambahan saat menjalankan container Docker. Untuk informasi selengkapnya, lihat berikut ini:
>
> - Gunakan interprocess communication dalam komponen kontainer Docker
> - Gunakan AWS kredensil dalam komponen wadah Docker (Linux)
> - Gunakan pengelola aliran di komponen wadah Docker (Linux)

3. Uji komponen untuk memverifikasi bahwa ia bekerja seperti yang diharapkan.

   > **Important**
   >
   > Anda harus memulai daemon Docker sebelum Anda men-deploy komponen.

   Setelah Anda men-deploy komponen lokal, Anda dapat menjalankan perintah kontainer docker ls untuk memverifikasi bahwa kontainer Anda berjalan.

   ```
   docker container ls
   ```

4. Saat komponen sudah siap, unggah komponen untuk digunakan AWS IoT Greengrass ke perangkat inti lainnya. Untuk informasi selengkapnya, lihat Publikasikan komponen untuk diterapkan ke perangkat inti Anda.

Jalankan kontainer Docker dari gambar privat di Amazon ECR

Bagian ini menjelaskan bagaimana Anda dapat membuat komponen kustom yang menjalankan kontainer Docker dari gambar Docker yang disimpan dalam repositori privat di Amazon ECR.

Untuk menjalankan kontainer Docker

1. Buat komponen khusus di perangkat AWS IoT Greengrass inti Anda. Gunakan contoh berikut resep, yang memiliki properti berikut:

- Komponen manajer aplikasi Docker sebagai dependensi. Komponen ini memungkinkan AWS IoT Greengrass untuk mengelola kredensial untuk mengunduh gambar dari repositori privat.
- Komponen layanan pertukaran token sebagai dependensi. Komponen ini memungkinkan AWS IoT Greengrass untuk mengambil AWS kredensil untuk berinteraksi dengan Amazon ECR.
- Artefak komponen yang menentukan gambar Docker dalam repositori Amazon ECR privat.
- Skrip lifecycle run yang menggunakan docker run untuk membuat dan memulai sebuah kontainer dari gambar.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.MyPrivateDockerComponent",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that runs a Docker container from a
 private Amazon ECR image.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.DockerApplicationManager": {
      "VersionRequirement": "~2.0.0"
    },
    "aws.greengrass.TokenExchangeService": {
      "VersionRequirement": "~2.0.0"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "all"
      },
      "Lifecycle": {
        "run": "docker run account-
id.dkr.ecr.region.amazonaws.com/repository[:tag|@digest]"
      },
      "Artifacts": [
        {
          "URI": "docker:account-
id.dkr.ecr.region.amazonaws.com/repository[:tag|@digest]"
        }
      ]
```

```
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.MyPrivateDockerComponent
ComponentVersion: '1.0.0'
ComponentDescription: 'A component that runs a Docker container from a private
 Amazon ECR image.'
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.DockerApplicationManager:
    VersionRequirement: ~2.0.0
  aws.greengrass.TokenExchangeService:
    VersionRequirement: ~2.0.0
Manifests:
  - Platform:
      os: all
    Lifecycle:
        run: docker run account-id.dkr.ecr.region.amazonaws.com/repository[:tag|
@digest]
    Artifacts:
      - URI: "docker:account-id.dkr.ecr.region.amazonaws.com/repository[:tag|
@digest]"
```

> **Note**
>
> Untuk menggunakan interprocess communication (IPC), AWS credentials, atau stream manager di komponen container Docker Anda, Anda harus menentukan opsi tambahan saat menjalankan container Docker. Untuk informasi selengkapnya, lihat berikut ini:
>
> - Gunakan interprocess communication dalam komponen kontainer Docker
> - Gunakan AWS kredensil dalam komponen wadah Docker (Linux)
> - Gunakan pengelola aliran di komponen wadah Docker (Linux)

2. Uji komponen untuk memverifikasi bahwa ia bekerja seperti yang diharapkan.

> **Important**
>
> Anda harus memulai daemon Docker sebelum Anda men-deploy komponen.

Setelah Anda men-deploy komponen lokal, Anda dapat menjalankan perintah kontainer docker ls untuk memverifikasi bahwa kontainer Anda berjalan.

```
docker container ls
```

3. Unggah komponen AWS IoT Greengrass untuk digunakan ke perangkat inti lainnya. Untuk informasi selengkapnya, lihat Publikasikan komponen untuk diterapkan ke perangkat inti Anda.

Jalankan kontainer Docker dari gambar di Amazon S3

Bagian ini menjelaskan bagaimana Anda dapat menjalankan kontainer Docker dalam komponen dari gambar Docker yang disimpan di Amazon S3.

Jalankan kontainer Docker dalam komponen dari gambar di Amazon S3

1. Jalankan perintah docker save untuk membuat cadangan kontainer Docker. Anda menyediakan cadangan ini sebagai artefak komponen untuk menjalankan kontainer pada AWS IoT Greengrass. Ganti *`hello-wrld`* dengan nama gambar, dan ganti *`hello-world.tar`* dengan nama file arsip yang akan dibuat.

   ```
   docker save hello-world > artifacts/com.example.MyDockerComponent/1.0.0/hello-world.tar
   ```

2. Buat komponen khusus di perangkat AWS IoT Greengrass inti Anda. Gunakan contoh resep berikut, yang memiliki properti berikut:

   - Skrip lifecycle install yang menggunakan docker load untuk memuat gambar Docker dari arsip.
   - Skrip lifecycle run yang menggunakan docker run untuk membuat dan memulai sebuah kontainer dari gambar. Opsi `--rm` membersihkan kontainer ketika keluar.

   JSON

   ```
   {
   ```

```
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.MyS3DockerComponent",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that runs a Docker container from an
 image in an S3 bucket.",
  "ComponentPublisher": "Amazon",
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": {
          "Script": "docker load -i {artifacts:path}/hello-world.tar"
        },
        "run": {
          "Script": "docker run --rm hello-world"
        }
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.MyS3DockerComponent
ComponentVersion: '1.0.0'
ComponentDescription: 'A component that runs a Docker container from an image in
 an S3 bucket.'
ComponentPublisher: Amazon
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install:
        Script: docker load -i {artifacts:path}/hello-world.tar
      run:
        Script: docker run --rm hello-world
```

> **Note**
>
> Untuk menggunakan interprocess communication (IPC), AWS credentials, atau stream manager di komponen container Docker Anda, Anda harus menentukan opsi tambahan saat menjalankan container Docker. Untuk informasi selengkapnya, lihat berikut ini:
>
> - Gunakan interprocess communication dalam komponen kontainer Docker
> - Gunakan AWS kredensil dalam komponen wadah Docker (Linux)
> - Gunakan pengelola aliran di komponen wadah Docker (Linux)

3. Uji komponen untuk memverifikasi bahwa ia bekerja seperti yang diharapkan.

   Setelah Anda men-deploy komponen lokal, Anda dapat menjalankan perintah kontainer docker ls untuk memverifikasi bahwa kontainer Anda berjalan.

   ```
   docker container ls
   ```

4. Ketika komponen sudah siap, unggah arsip gambar Docker ke bucket S3, dan tambahkan URI ke resep komponen. Kemudian, Anda dapat mengunggah komponen AWS IoT Greengrass untuk menyebarkan ke perangkat inti lainnya. Untuk informasi selengkapnya, lihat Publikasikan komponen untuk diterapkan ke perangkat inti Anda.

   Setelah selesai, resep komponen akan terlihat seperti contoh berikut.

   JSON

   ```
   {
     "RecipeFormatVersion": "2020-01-25",
     "ComponentName": "com.example.MyS3DockerComponent",
     "ComponentVersion": "1.0.0",
     "ComponentDescription": "A component that runs a Docker container from an
    image in an S3 bucket.",
     "ComponentPublisher": "Amazon",
     "Manifests": [
       {
         "Platform": {
           "os": "linux"
         },
         "Lifecycle": {
   ```

```
        "install": {
          "Script": "docker load -i {artifacts:path}/hello-world.tar"
        },
        "run": {
          "Script": "docker run --rm hello-world"
        }
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.MyDockerComponent/1.0.0/hello-world.tar"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.MyS3DockerComponent
ComponentVersion: '1.0.0'
ComponentDescription: 'A component that runs a Docker container from an image in
 an S3 bucket.'
ComponentPublisher: Amazon
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install:
        Script: docker load -i {artifacts:path}/hello-world.tar
      run:
        Script: docker run --rm hello-world
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.MyDockerComponent/1.0.0/hello-world.tar
```

Gunakan interprocess communication dalam komponen kontainer Docker

Anda dapat menggunakan perpustakaan Greengrass interprocess communication (IPC) AWS IoT Device SDK untuk berkomunikasi dengan inti Greengrass, komponen Greengrass lainnya, dan. AWS

IoT Core Untuk informasi selengkapnya, lihat Gunakan AWS IoT Device SDK untuk berkomunikasi dengan inti Greengrass, komponen lain, dan AWS IoT Core.

Untuk menggunakan IPC dalam komponen kontainer Docker, Anda harus menjalankan wadah Docker dengan parameter berikut:

- Pasang soket IPC di wadah. Inti Greengrass menyediakan jalur file soket IPC dalam variabel lingkungan. `AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT`
- Atur variabel `SVCUID` dan `AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT` lingkungan ke nilai yang disediakan oleh inti Greengrass ke komponen. Komponen Anda menggunakan variabel lingkungan ini untuk mengautentikasi koneksi ke inti Greengrass.

Example Contoh resep: Publikasikan pesan MQTT ke (Python) AWS IoT Core

Resep berikut mendefinisikan contoh komponen kontainer Docker yang menerbitkan pesan MQTT ke. AWS IoT Core Resep ini memiliki sifat sebagai berikut:

- Kebijakan otorisasi (`accessControl`) yang memungkinkan komponen untuk mempublikasikan pesan MQTT pada semua topik. AWS IoT Core Untuk informasi lebih lanjut, lihat Otorisasi komponen untuk melakukan operasi IPC dan otorisasi AWS IoT Core MQTT IPC.
- Artefak komponen yang menentukan image Docker sebagai arsip TAR di Amazon S3.
- Skrip penginstalan siklus hidup yang memuat gambar Docker dari arsip TAR.
- Skrip proses siklus hidup yang menjalankan wadah Docker dari gambar. Perintah Docker run memiliki argumen berikut:
  - `-v`Argumen memasang soket Greengrass IPC di wadah.
  - Dua `-e` argumen pertama mengatur variabel lingkungan yang diperlukan dalam wadah Docker.
  - `-e`Argumen tambahan mengatur variabel lingkungan yang digunakan oleh contoh ini.
  - `--rm`Argumen membersihkan wadah saat keluar.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.python.docker.PublishToIoTCore",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Uses interprocess communication to publish an MQTT
 message to IoT Core.",
```

```
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "topic": "test/topic/java",
      "message": "Hello, World!",
      "qos": "1",
      "accessControl": {
        "aws.greengrass.ipc.mqttproxy": {
          "com.example.python.docker.PublishToIoTCore:pubsub:1": {
            "policyDescription": "Allows access to publish to IoT Core on all
 topics.",
            "operations": [
              "aws.greengrass#PublishToIoTCore"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "all"
      },
      "Lifecycle": {
        "install": "docker load -i {artifacts:path}/publish-to-iot-core.tar",
        "run": "docker run -v $AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT:
$AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT -e SVCUID -e
 AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT -e MQTT_TOPIC=
\"{configuration:/topic}\" -e MQTT_MESSAGE=\"{configuration:/message}\" -e MQTT_QOS=
\"{configuration:/qos}\" --rm publish-to-iot-core"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.PublishToIoTCore/1.0.0/publish-to-iot-core.tar"
        }
      ]
    }
  ]
```

```
}
```

YAML

```
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.python.docker.PublishToIoTCore
ComponentVersion: 1.0.0
ComponentDescription: Uses interprocess communication to publish an MQTT message to
 IoT Core.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    topic: 'test/topic/java'
    message: 'Hello, World!'
    qos: '1'
    accessControl:
      aws.greengrass.ipc.mqttproxy:
        'com.example.python.docker.PublishToIoTCore:pubsub:1':
          policyDescription: Allows access to publish to IoT Core on all topics.
          operations:
            - 'aws.greengrass#PublishToIoTCore'
          resources:
            - '*'
Manifests:
  - Platform:
      os: all
    Lifecycle:
      install: 'docker load -i {artifacts:path}/publish-to-iot-core.tar'
      run: |
        docker run \
          -v $AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT:
$AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT \
          -e SVCUID \
          -e AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT \
          -e MQTT_TOPIC="{configuration:/topic}" \
          -e MQTT_MESSAGE="{configuration:/message}" \
          -e MQTT_QOS="{configuration:/qos}" \
          --rm publish-to-iot-core
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.PublishToIoTCore/1.0.0/publish-to-iot-core.tar
```

Gunakan AWS kredensil dalam komponen wadah Docker (Linux)

Anda dapat menggunakan komponen layanan pertukaran token untuk berinteraksi dengan AWS layanan di komponen Greengrass. Komponen ini menyediakan AWS kredensyal dari peran pertukaran token perangkat inti menggunakan server kontainer lokal. Untuk informasi selengkapnya, lihat Berinteraksilah dengan layanan AWS.

> **Note**
>
> Contoh di bagian ini hanya berfungsi pada perangkat inti Linux.

Untuk menggunakan AWS kredensyal dari layanan pertukaran token dalam komponen kontainer Docker, Anda harus menjalankan container Docker dengan parameter berikut:

- Berikan akses ke jaringan host menggunakan `--network=host` argumen. Opsi ini memungkinkan kontainer Docker untuk terhubung ke layanan pertukaran token lokal untuk mengambil kredensyal AWS . Argumen ini hanya berfungsi pada Docker untuk Linux.

  > **Warning**
  >
  > Opsi ini memberikan akses kontainer ke semua antarmuka jaringan lokal pada host, jadi opsi ini kurang aman daripada jika Anda menjalankan kontainer Docker tanpa akses ini ke jaringan host. Pertimbangkan ini saat Anda mengembangkan dan menjalankan komponen kontainer Docker yang menggunakan opsi ini. Untuk informasi selengkapnya, lihat Jaringan: host di Dokumentasi Docker.

- Atur variabel `AWS_CONTAINER_CREDENTIALS_FULL_URI` dan `AWS_CONTAINER_AUTHORIZATION_TOKEN` lingkungan ke nilai yang disediakan oleh inti Greengrass ke komponen. AWS SDK menggunakan variabel lingkungan ini untuk mengambil kredensil AWS .

Example Contoh resep: Daftar bucket S3 dalam komponen wadah Docker (Python)

Resep berikut mendefinisikan contoh komponen kontainer Docker yang mencantumkan bucket S3 di bucket Anda. Akun AWS Resep ini memiliki sifat sebagai berikut:

- Komponen layanan pertukaran token sebagai dependensi. Ketergantungan ini memungkinkan komponen untuk mengambil AWS kredensi untuk berinteraksi dengan layanan lain. AWS

- Artefak komponen yang menentukan image Docker sebagai arsip tar di Amazon S3.
- Skrip penginstalan siklus hidup yang memuat gambar Docker dari arsip TAR.
- Skrip proses siklus hidup yang menjalankan wadah Docker dari gambar. Perintah Docker run memiliki argumen berikut:
  - `--network=host`Argumen menyediakan akses kontainer ke jaringan host, sehingga wadah dapat terhubung ke layanan pertukaran token.
  - `-e`Argumen menetapkan variabel lingkungan yang diperlukan dalam wadah Docker.
  - `--rm`Argumen membersihkan wadah saat keluar.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.python.docker.ListS3Buckets",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Uses the token exchange service to lists your S3
 buckets.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.TokenExchangeService": {
      "VersionRequirement": "^2.0.0",
      "DependencyType": "HARD"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "docker load -i {artifacts:path}/list-s3-buckets.tar",
        "run": "docker run --network=host -e AWS_CONTAINER_AUTHORIZATION_TOKEN -e
 AWS_CONTAINER_CREDENTIALS_FULL_URI --rm list-s3-buckets"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.ListS3Buckets/1.0.0/list-s3-buckets.tar"
        }
      ]
```

```
    }
  ]
}
```

YAML

```
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.python.docker.ListS3Buckets
ComponentVersion: 1.0.0
ComponentDescription: Uses the token exchange service to lists your S3 buckets.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.TokenExchangeService:
    VersionRequirement: ^2.0.0
    DependencyType: HARD
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: 'docker load -i {artifacts:path}/list-s3-buckets.tar'
      run: |
        docker run \
          --network=host \
          -e AWS_CONTAINER_AUTHORIZATION_TOKEN \
          -e AWS_CONTAINER_CREDENTIALS_FULL_URI \
          --rm list-s3-buckets
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.ListS3Buckets/1.0.0/list-s3-buckets.tar
```

Gunakan pengelola aliran di komponen wadah Docker (Linux)

Anda dapat menggunakan komponen pengelola aliran untuk mengelola aliran data di komponen Greengrass. Komponen ini memungkinkan Anda untuk memproses aliran data dan mentransfer data IoT volume tinggi ke file. AWS Cloud AWS IoT Greengrass menyediakan SDK manajer aliran yang Anda gunakan untuk berinteraksi dengan komponen pengelola aliran. Untuk informasi selengkapnya, lihat Kelola aliran data di perangkat inti Greengrass.

> **Note**
>
> Contoh di bagian ini hanya berfungsi pada perangkat inti Linux.

Untuk menggunakan SDK manajer aliran dalam komponen kontainer Docker, Anda harus menjalankan container Docker dengan parameter berikut:

- Berikan akses ke jaringan host menggunakan `--network=host` argumen. Opsi ini memungkinkan wadah Docker untuk berinteraksi dengan komponen pengelola aliran melalui koneksi TLS lokal. Argumen ini hanya berfungsi pada Docker untuk Linux

  > **Warning**
  >
  > Opsi ini memberikan akses kontainer ke semua antarmuka jaringan lokal pada host, jadi opsi ini kurang aman daripada jika Anda menjalankan kontainer Docker tanpa akses ini ke jaringan host. Pertimbangkan ini saat Anda mengembangkan dan menjalankan komponen kontainer Docker yang menggunakan opsi ini. Untuk informasi selengkapnya, lihat Jaringan: host di Dokumentasi Docker.

- Jika Anda mengonfigurasi komponen pengelola aliran agar memerlukan otentikasi, yang merupakan perilaku default, setel variabel `AWS_CONTAINER_CREDENTIALS_FULL_URI` lingkungan ke nilai yang disediakan inti Greengrass ke komponen. Untuk informasi selengkapnya, lihat konfigurasi manajer aliran.
- Jika Anda mengonfigurasi komponen pengelola aliran untuk menggunakan port non-default, gunakan komunikasi antarproses (IPC) untuk mendapatkan port dari konfigurasi komponen manajer aliran. Anda harus menjalankan wadah Docker dengan opsi tambahan untuk menggunakan IPC. Untuk informasi selengkapnya, lihat berikut ini:
  - Hubungkan ke manajer pengaliran dalam kode aplikasi
  - Gunakan interprocess communication dalam komponen kontainer Docker

Example Contoh resep: Streaming file ke bucket S3 dalam komponen wadah Docker (Python)

Resep berikut mendefinisikan contoh komponen kontainer Docker yang membuat file dan mengalirkannya ke bucket S3. Resep ini memiliki sifat sebagai berikut:

- Komponen stream manager sebagai dependensi. Ketergantungan ini memungkinkan komponen menggunakan SDK manajer aliran untuk berinteraksi dengan komponen pengelola aliran.
- Artefak komponen yang menentukan image Docker sebagai arsip TAR di Amazon S3.
- Skrip penginstalan siklus hidup yang memuat gambar Docker dari arsip TAR.
- Skrip proses siklus hidup yang menjalankan wadah Docker dari gambar. Perintah Docker run memiliki argumen berikut:
  - `--network=host`Argumen menyediakan akses kontainer ke jaringan host, sehingga kontainer dapat terhubung ke komponen manajer aliran.
  - `-e`Argumen pertama menetapkan variabel `AWS_CONTAINER_AUTHORIZATION_TOKEN` lingkungan yang diperlukan dalam wadah Docker.
  - `-e`Argumen tambahan mengatur variabel lingkungan yang digunakan oleh contoh ini.
  - `-v`Argumen memasang folder kerja komponen di wadah. Contoh ini membuat file di folder kerja untuk mengunggah file itu ke Amazon S3 menggunakan pengelola aliran.
  - `--rm`Argumen membersihkan wadah saat keluar.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.python.docker.StreamFileToS3",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Creates a text file and uses stream manager to stream the
 file to S3.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.StreamManager": {
      "VersionRequirement": "^2.0.0",
      "DependencyType": "HARD"
    }
  },
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "bucketName": ""
    }
  },
  "Manifests": [
    {
      "Platform": {
```

```
        "os": "linux"
      },
      "Lifecycle": {
        "install": "docker load -i {artifacts:path}/stream-file-to-s3.tar",
        "run": "docker run --network=host -e AWS_CONTAINER_AUTHORIZATION_TOKEN
 -e BUCKET_NAME=\"{configuration:/bucketName}\" -e WORK_PATH=\"{work:path}\" -v
 {work:path}:{work:path} --rm stream-file-to-s3"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.StreamFileToS3/1.0.0/stream-file-to-s3.tar"
        }
      ]
    }
  ]
}
```

YAML

```
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.python.docker.StreamFileToS3
ComponentVersion: 1.0.0
ComponentDescription: Creates a text file and uses stream manager to stream the file
 to S3.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.StreamManager:
    VersionRequirement: ^2.0.0
    DependencyType: HARD
ComponentConfiguration:
  DefaultConfiguration:
    bucketName: ''
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: 'docker load -i {artifacts:path}/stream-file-to-s3.tar'
      run: |
        docker run \
          --network=host \
          -e AWS_CONTAINER_AUTHORIZATION_TOKEN \
          -e BUCKET_NAME="{configuration:/bucketName}" \
```

```
            -e WORK_PATH="{work:path}" \
            -v {work:path}:{work:path} \
            --rm stream-file-to-s3
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.StreamFileToS3/1.0.0/stream-file-to-s3.tar
```

AWS IoT Greengrass referensi resep komponen

Komponen resep adalah file yang mendefinisikan komponen detail, dependensi, artefak, dan siklus hidup. Siklus aktif komponen menentukan perintah yang akan dijalankan untuk menginstal, menjalankan, dan menutup komponen, misalnya. AWS IoT Greengrass Inti menggunakan siklus hidup yang Anda tentukan dalam resep untuk menginstal dan menjalankan komponen. AWS IoT Greengrass Layanan menggunakan resep untuk mengidentifikasi dependensi dan artefak untuk diterapkan ke perangkat inti Anda saat Anda menerapkan komponen.

Dalam resep itu, Anda dapat menentukan dependensi dan siklus hidup yang unik untuk setiap platform yang didukung komponen. Anda dapat menggunakan kemampuan ini untuk men-deploy komponen ke perangkat dengan beberapa platform yang memiliki persyaratan yang berbeda. Anda juga dapat menggunakan ini untuk AWS IoT Greengrass mencegah menginstal komponen pada perangkat yang tidak mendukungnya.

Setiap resep berisi daftar manifes. Setiap manifes menentukan seperangkat persyaratan platform dan siklus hidup dan artefak yang akan digunakan untuk perangkat inti yang platformnya memenuhi persyaratan tersebut. Perangkat inti menggunakan manifes pertama dengan persyaratan platform yang dipenuhi oleh perangkat. Tentukan manifes tanpa persyaratan platform apa pun agar cocok dengan setiap perangkat inti.

Anda juga dapat menentukan siklus hidup global yang tidak ada dalam manifes. Dalam siklus hidup global, Anda dapat menggunakan tombol pilihan yang mengidentifikasi sub-bagian dari siklus hidup. Kemudian, Anda dapat menentukan kunci pilihan ini dalam manifes untuk menggunakan bagian dari siklus hidup global selain siklus hidup manifes. Perangkat inti hanya menggunakan tombol pemilihan manifes jika manifes tidak menentukan siklus hidup. Anda dapat menggunakan pilihan `all` dalam manifes untuk mencocokkan bagian siklus hidup global tanpa kunci seleksi.

Setelah perangkat lunak AWS IoT Greengrass Core memilih manifes yang cocok dengan perangkat inti, perangkat tersebut melakukan hal berikut untuk mengidentifikasi langkah-langkah siklus hidup yang akan digunakan:

- Jika manifes yang dipilih menentukan siklus hidup, perangkat inti akan menggunakan siklus hidup tersebut.
- Jika manifes yang dipilih tidak menentukan siklus hidup, perangkat inti akan menggunakan siklus hidup global. Perangkat inti melakukan hal berikut untuk mengidentifikasi bagian siklus hidup global mana yang akan digunakan:
  - Jika manifes mendefinisikan kunci pilihan, perangkat inti akan menggunakan bagian siklus hidup global yang berisi kunci pilihan manifes.
  - Jika manifes tidak mendefinisikan kunci pilihan, perangkat inti akan menggunakan bagian siklus hidup global yang tidak memiliki kunci pilihan. Perilaku ini setara dengan manifes yang menentukan pilihan `all`.

> **Important**
>
> Sebuah perangkat inti harus cocok setidaknya dengan satu persyaratan platform manifes untuk menginstal komponen. Jika tidak ada manifes yang cocok dengan perangkat AWS IoT Greengrass inti, maka perangkat lunak Core tidak menginstal komponen dan penerapan gagal.

Anda bisa menentukan resep dalam format JSON atau YAML. Bagian contoh resep mencakup resep dalam setiap format.

Topik

Validasi resep

Greengrass memvalidasi resep komponen JSON atau YAMG saat membuat versi komponen. Validasi resep ini memeriksa resep komponen JSON atau YAMAL Anda untuk kesalahan umum untuk mencegah potensi masalah penerapan. Validasi memeriksa resep untuk kesalahan umum (misalnya, koma, tanda kurung gigi, dan bidang yang hilang) dan untuk memastikan resepnya terbentuk dengan baik.

Jika Anda menerima pesan kesalahan validasi resep, periksa resep Anda apakah ada koma, tanda kurung gigi, atau bidang yang hilang. Verifikasi bahwa Anda tidak melewatkan bidang apa pun dengan melihat format resep.

Format resep

Bila Anda menentukan resep untuk komponen, Anda menentukan informasi berikut dalam dokumen resep. Struktur yang sama berlaku untuk resep dalam format YAML dan JSON.

`RecipeFormatVersion`

Versi templat untuk resep itu. Pilih opsi berikut:

- `2020-01-25`

`ComponentName`

Nama komponen yang ditentukan oleh resep ini. Nama komponen harus unik di Anda Akun AWS di setiap Wilayah.

> **Kiat**
>
> - Gunakan format nama domain terbalik untuk menghindari tabrakan nama dalam perusahaan Anda. Misalnya, jika perusahaan Anda memiliki `example.com` dan Anda mengerjakan proyek energi matahari, Anda dapat menamai komponen Hello World Anda `com.example.solar.HelloWorld`. Hal ini membantu menghindari tabrakan nama komponen dalam perusahaan Anda.
> - Hindari awalan `aws.greengrass` dalam nama komponen Anda. AWS IoT Greengrass menggunakan awalan ini untuk Komponen publik yang disediakannya. Jika Anda memilih nama yang sama sebagai komponen publik, komponen Anda menggantikan komponen tersebut. Kemudian, AWS IoT Greengrass sediakan komponen Anda alih-alih komponen publik saat menyebarkan komponen dengan ketergantungan pada komponen publik tersebut. Fitur ini memungkinkan Anda untuk menimpa perilaku komponen publik, tetapi juga dapat merusak komponen lain jika Anda tidak berniat untuk menimpa komponen publik.

`ComponentVersion`

Versi komponen. Nilai maksimum untuk nilai mayor, minor, dan patch adalah 999999.

> **Note**
>
> AWS IoT Greengrass menggunakan versi semantik untuk komponen. Versi semantik mengikuti sistem nomor mayor.minor.patch. Sebagai contoh, versi `1.0.0` merupakan rilis mayor pertama untuk sebuah komponen. Untuk informasi lebih lanjut, lihat spesifikasi versi semantik.

`ComponentDescription`

(Opsional) Deskripsi komponen.

`ComponentPublisher`

Penerbit atau penulis komponen.

`ComponentConfiguration`

(Opsional) Sebuah objek yang menentukan konfigurasi atau parameter untuk komponen. Anda menentukan konfigurasi default, dan kemudian ketika Anda men-deploy komponen, Anda dapat menentukan objek konfigurasi yang akan disediakan ke komponen. Konfigurasi komponen mendukung parameter bersusun dan struktur. Objek ini berisi informasi berikut:

`DefaultConfiguration`

Objek yang menentukan konfigurasi default untuk komponen. Anda menentukan struktur objek ini.

> **Note**
>
> AWS IoT Greengrass menggunakan JSON untuk nilai konfigurasi. JSON menentukan jenis nomor tetapi tidak membedakan antara bilangan bulat dan float. Akibatnya, nilai konfigurasi mungkin berubah menjadi float di AWS IoT Greengrass. Untuk memastikan bahwa komponen Anda menggunakan jenis data yang benar, kami sarankan Anda menentukan nilai konfigurasi numerik sebagai string. Kemudian, buat komponen Anda mengurainya sebagai bilangan bulat atau float. Hal ini memastikan bahwa nilai konfigurasi Anda memiliki tipe yang sama dalam konfigurasi dan pada perangkat inti Anda.

`ComponentDependencies`

(Opsional) Sebuah kamus objek yang masing-masing mendefinisikan dependensi komponen untuk komponen tersebut. Kunci untuk setiap objek mengidentifikasi nama ketergantungan komponen. AWS IoT Greengrass menginstal dependensi komponen saat komponen diinstal. AWS IoT Greengrass menunggu dependensi dimulai sebelum memulai komponen. Setiap objek berisi informasi berikut.

`VersionRequirement`

Kendala versi semantik npm-style yang menentukan versi komponen yang kompatibel untuk dependensi ini. Anda dapat menentukan versi atau rentang versi. Untuk informasi lebih lanjut, lihat kalkulator versi semantik npm.

`DependencyType`

(Opsional) Jenis dependensi ini. Pilih salah satu dari opsi berikut:

- `SOFT` — Komponen tidak me-restart jika dependensi mengubah keadaan.
- `HARD` — Komponen me-restart jika dependensi mengubah keadaan.

Default ke `HARD`.

`ComponentType`

(Opsional) Jenis komponen.

> **Note**
>
> Kami tidak menyarankan Anda menentukan jenis komponen dalam resep. AWS IoT Greengrass menetapkan tipe untuk Anda saat Anda membuat komponen.

Jenis ini dapat berupa salah satu dari jenis-jenis berikut:

- `aws.greengrass.generic` — Komponen menjalankan perintah atau menyediakan artefak.
- `aws.greengrass.lambda` – Komponen menjalankan fungsi Lambda menggunakan Komponen peluncur Lambda. Parameter `ComponentSource` menentukan ARN fungsi Lambda yang dijalankan oleh komponen ini.

  Kami tidak menyarankan Anda menggunakan opsi ini, karena opsi ini diatur oleh AWS IoT Greengrass saat Anda membuat komponen dari fungsi Lambda. Untuk informasi selengkapnya, lihat Jalankan fungsi AWS Lambda.

- `aws.greengrass.plugin` – Komponen ini berjalan pada komponen plugin dalam Java Virtual Machine (JVM) yang sama seperti inti Greengrass. Jika Anda men-deploy atau me-restart komponen plugin, inti Greengrass akan me-restart.

  Komponen plugin menggunakan file log yang sama seperti inti Greengrass. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

  Kami tidak menyarankan Anda menggunakan opsi ini dalam resep komponen, karena ini ditujukan untuk komponen yang AWS disediakan yang ditulis di Java yang secara langsung berinteraksi dengan inti Greengrass. Untuk informasi lebih lanjut tentang komponen publik mana yang merupakan plugin, lihat Komponen yang disediakan oleh AWS.
- `aws.greengrass.nucleus` — Komponen nukleus. Untuk informasi selengkapnya, lihat Inti Greengrass.

  Kami tidak menyarankan Anda menggunakan opsi ini dalam resep komponen. Opsi ini ditujukan untuk komponen inti Greengrass, yang menyediakan fungsionalitas minimum perangkat lunak inti AWS IoT Greengrass .

Default-nya adalah `aws.greengrass.generic` saat Anda membuat komponen dari resep, atau `aws.greengrass.lambda` saat Anda membuat komponen dari fungsi Lambda.

Untuk informasi selengkapnya, lihat Jenis komponen.

`ComponentSource`

(Opsional) ARN fungsi Lambda yang dijalankan komponen.

Kami tidak menyarankan Anda menentukan sumber komponen dalam resep. AWS IoT Greengrass menetapkan parameter ini untuk Anda saat Anda membuat komponen dari fungsi Lambda. Untuk informasi selengkapnya, lihat Jalankan fungsi AWS Lambda.

`Manifests`

Daftar objek yang masing-masing menentukan siklus hidup, parameter, dan persyaratan komponen untuk platform. Jika perangkat inti cocok dengan persyaratan platform beberapa manifes, AWS IoT Greengrass gunakan manifes pertama yang cocok dengan perangkat inti. Untuk memastikan bahwa perangkat inti menggunakan manifes yang benar, tentukan manifes dengan persyaratan platform ketat terlebih dahulu. Manifes yang berlaku untuk semua platform harus menjadi manifes terakhir dalam daftar.

> **Important**
>
> Sebuah perangkat inti harus cocok setidaknya dengan satu persyaratan platform manifes untuk menginstal komponen. Jika tidak ada manifes yang cocok dengan perangkat AWS IoT Greengrass inti, maka perangkat lunak Core tidak menginstal komponen dan penerapan gagal.

Setiap objek berisi informasi berikut:

`Name`

(Opsional) Nama yang bersahabat untuk platform yang ditentukan leh manifes ini.

Jika Anda menghilangkan parameter ini, AWS IoT Greengrass buat nama dari platform `os` dan`architecture`.

`Platform`

(Opsional) Sebuah objek yang menentukan platform yang padanya manifes ini berlaku. Hilangkan parameter ini untuk menentukan manifes yang berlaku untuk semua platform.

Objek ini menentukan pasangan kunci-nilai tentang platform di mana perangkat inti berjalan. Saat Anda menerapkan komponen ini, perangkat lunak AWS IoT Greengrass Core membandingkan pasangan nilai kunci ini dengan atribut platform pada perangkat inti. Perangkat lunak AWS IoT Greengrass Core selalu mendefinisikan `os` dan`architecture`, dan mungkin mendefinisikan atribut tambahan. Anda dapat menentukan atribut platform kustom untuk perangkat inti ketika Anda men-deploy komponen inti Greengrass. Untuk informasi lebih lanjut, lihat bagian platform menimpa parameter dari komponen nukleus Greengrass.

Untuk setiap pasangan kunci-nilai, Anda dapat menentukan salah satu nilai berikut:

- Nilai yang tepat, seperti `linux` atau`windows`. Nilai yang tepat harus dimulai dengan huruf atau angka.
- `*`, yang cocok dengan nilai apa pun. Hal ini juga cocok ketika nilai tidak ada.
- Sebuah ekspresi reguler gaya Java, seperti `/windows|linux/`. Ekspresi reguler harus dimulai dan diakhiri dengan karakter garis miring (`/`). Misalnya, ekspresi reguler `/.+/` cocok dengan nilai non-kosong.

Objek ini berisi informasi berikut:

`os`

(Opsional) Nama sistem operasi untuk platform yang didukung oleh manifes ini. Platform umum mencakup nilai-nilai berikut:

- `linux`
- `windows`
- `darwin` (macOS)

`architecture`

(Opsional) Arsitektur prosesor untuk platform yang didukung oleh manifes ini. Arsitektur umum mencakup nilai-nilai berikut:

- `amd64`
- `arm`
- `aarch64`
- `x86`

`architecture.detail`

(Opsional) Detail arsitektur prosesor untuk platform yang didukung manifes ini. Rincian arsitektur umum mencakup nilai-nilai berikut:

- `arm61`
- `arm71`
- `arm81`

`key`

(Opsional) Atribut platform yang Anda tetapkan untuk manifes ini. Ganti *`kunci`* dengan nama atribut platform. Perangkat lunak AWS IoT Greengrass Core mencocokkan atribut platform ini dengan pasangan kunci-nilai yang Anda tentukan dalam konfigurasi komponen inti Greengrass. Untuk informasi lebih lanjut, lihat bagian platform menimpa parameter dari komponen nukleus Greengrass.

> **Tip**
>
> Gunakan format nama domain terbalik untuk menghindari tabrakan nama dalam perusahaan Anda. Misalnya, jika perusahaan Anda memiliki `example.com` dan Anda mengerjakan proyek radio, Anda dapat memberi nama atribut platform

kustom `com.example.radio.RadioModule`. Hal ini membantu menghindari tabrakan nama atribut platform dalam perusahaan Anda.

Misalnya, Anda dapat menentukan atribut platform, `com.example.radio.RadioModule`, untuk menentukan manifes yang berbeda berdasarkan modul radio yang tersedia pada perangkat inti. Setiap manifes dapat mencakup artefak yang berbeda yang berlaku pada konfigurasi perangkat keras yang berbeda, agar Anda men-deploy serangkaian perangkat lunak minimal ke perangkat inti.

`Lifecycle`

Objek atau string yang mendefinisikan cara menginstal dan menjalankan komponen pada platform yang didefinisikan oleh manifes ini. Anda juga dapat menentukan siklus hidup global yang berlaku untuk semua platform. Perangkat inti hanya menggunakan siklus hidup global jika manifes yang akan digunakan tidak menentukan siklus hidup.

> **Note**
>
> Anda mendefinisikan siklus hidup ini dalam manifes. Langkah-langkah siklus hidup yang Anda tentukan di sini hanya berlaku untuk platform yang ditentukan oleh manifes ini. Anda juga dapat menentukan siklus hidup global yang berlaku untuk semua platform.

Objek atau string ini berisi informasi berikut:

`Setenv`

(Opsional) Kamus variabel lingkungan untuk menyediakan semua skrip siklus hidup. Anda dapat mengganti variabel lingkungan ini dengan `Setenv` dalam setiap skrip siklus hidup.

`install`

(Opsional) Objek atau string yang mendefinisikan skrip untuk dijalankan ketika komponen diinstal. Perangkat lunak AWS IoT Greengrass Core juga menjalankan langkah siklus hidup ini setiap kali perangkat lunak diluncurkan.

Jika skrip `install` keluar dengan kode sukses, komponen tersebut akan memasuki keadaan `INSTALLED`.

Objek atau string ini berisi informasi berikut:

`Script`

Skrip yang akan dijalankan.

`RequiresPrivilege`

(Opsional) Anda dapat menjalankan skrip dengan hak istimewa root. Jika Anda menyetel opsi ini`true`, maka perangkat lunak AWS IoT Greengrass Core menjalankan skrip siklus hidup ini sebagai root, bukan sebagai pengguna sistem yang Anda konfigurasikan untuk menjalankan komponen ini. Default ke `false`.

`Skipif`

(Opsional) Periksa untuk menentukan apakah akan menjalankan skrip atau tidak. Anda dapat menentukan untuk memeriksa apakah executable berada di jalur atau apakah file ada. Jika output benar, maka Perangkat lunak inti AWS IoT Greengrass melewati langkah itu. Pilih salah satu cek berikut:

- `onpath` *`runnable`* — Periksa apakah runnable berada di jalur sistem. Misalnya, gunakan **`onpath python3`** untuk melewati langkah siklus hidup ini jika Python 3 tersedia.
- `exists` *`file`* — Periksa apakah file ada. Misalnya, gunakan **`exists /tmp/my-configuration.db`** untuk melewati langkah siklus hidup ini jika `/tmp/my-configuration.db` ada.

`Timeout`

(Opsional) Jumlah maksimum waktu dalam detik yang dapat dijalankan skrip sebelum perangkat lunak inti AWS IoT Greengrass mengakhiri proses.

Default: 120 detik.

`Setenv`

(Opsional) Kamus variabel lingkungan untuk menyediakan skrip. Variabel lingkungan ini menimpa variabel yang Anda berikan di `Lifecycle.Setenv`.

`run`

(Opsional) Objek atau string yang mendefinisikan skrip untuk dijalankan ketika komponen dimulai.

Komponen memasuki keadaan `RUNNING` saat langkah siklus hidup ini berjalan. Jika skrip `run` keluar dengan kode sukses, komponen tersebut akan memasuki keadaan `STOPPING`.

Jika `shutdown` skrip ditentukan, itu berjalan; jika tidak komponen memasuki `FINISHED` status.

Komponen yang bergantung pada komponen ini dimulai saat langkah siklus hidup ini berjalan. Untuk menjalankan proses latar belakang, seperti layanan yang digunakan komponen dependen, gunakan langkah siklus hidup `startup` sebagai gantinya.

Saat Anda menerapkan komponen dengan `run` siklus hidup, perangkat inti dapat melaporkan penerapan selengkap segera setelah skrip siklus hidup ini berjalan. Akibatnya, penerapan dapat selesai dan berhasil bahkan jika skrip `run` siklus hidup gagal segera setelah dijalankan. Jika Anda ingin status penerapan bergantung pada hasil skrip awal komponen, gunakan langkah `startup` siklus hidup sebagai gantinya.

> **Note**
>
> Anda dapat menentukan hanya satu `startup` atau `run` siklus hidup.

Objek atau string ini berisi informasi berikut:

`Script`

Skrip yang akan dijalankan.

`RequiresPrivilege`

(Opsional) Anda dapat menjalankan skrip dengan hak istimewa root. Jika Anda menyetel opsi ini`true`, maka perangkat lunak AWS IoT Greengrass Core menjalankan skrip siklus hidup ini sebagai root, bukan sebagai pengguna sistem yang Anda konfigurasikan untuk menjalankan komponen ini. Default ke `false`.

`Skipif`

(Opsional) Periksa untuk menentukan apakah akan menjalankan skrip atau tidak. Anda dapat menentukan untuk memeriksa apakah executable berada di jalur atau apakah file ada. Jika output benar, maka Perangkat lunak inti AWS IoT Greengrass melewati langkah itu. Pilih salah satu cek berikut:

- `onpath` *`runnable`* — Periksa apakah runnable berada di jalur sistem. Misalnya, gunakan **`onpath python3`** untuk melewati langkah siklus hidup ini jika Python 3 tersedia.

- `exists` *`file`* — Periksa apakah file ada. Misalnya, gunakan **`exists /tmp/my-configuration.db`** untuk melewati langkah siklus hidup ini jika `/tmp/my-configuration.db` ada.

`Timeout`

(Opsional) Jumlah waktu maksimum dalam hitungan detik skrip dapat dijalankan sebelum perangkat lunak AWS IoT Greengrass Inti menghentikan proses.

Langkah siklus hidup ini tidak berbatas waktu secara default. Jika Anda menghilangkan batas waktu ini, skrip `run` akan berjalan sampai skrip tersebut keluar.

`Setenv`

(Opsional) Kamus variabel lingkungan untuk menyediakan skrip. Variabel lingkungan ini menimpa variabel yang Anda berikan di `Lifecycle.Setenv`.

`startup`

(Opsional) Objek atau string yang mendefinisikan proses latar belakang untuk dijalankan ketika komponen dimulai.

Gunakan `startup` untuk menjalankan perintah yang harus keluar dengan sukses atau memperbarui status komponen `RUNNING` sebelum komponen dependen dapat dimulai. Gunakan operasi UpdateStateIPC untuk mengatur status komponen ke `RUNNING` atau `ERRORED` ketika komponen memulai skrip yang tidak keluar. Misalnya, Anda dapat menentukan langkah `startup` yang memulai proses MySQL dengan `/etc/init.d/mysqld start`.

Komponen memasuki keadaan `STARTING` saat langkah siklus hidup ini berjalan. Jika skrip `startup` keluar dengan kode sukses, komponen tersebut akan memasuki keadaan `RUNNING`. Kemudian, komponen dependen bisa dimulai.

Saat Anda menerapkan komponen dengan `startup` siklus hidup, perangkat inti dapat melaporkan penerapan sebagai selesai setelah skrip siklus hidup ini keluar atau melaporkan statusnya. Dengan kata lain, status penerapan adalah `IN_PROGRESS` sampai semua skrip startup komponen keluar atau melaporkan status.

> **Note**
>
> Anda dapat menentukan hanya satu `startup` atau `run` siklus hidup.

Objek atau string ini berisi informasi berikut:

`Script`

Skrip yang akan dijalankan.

`RequiresPrivilege`

(Opsional) Anda dapat menjalankan skrip dengan hak istimewa root. Jika Anda menyetel opsi ini`true`, maka perangkat lunak AWS IoT Greengrass Core menjalankan skrip siklus hidup ini sebagai root, bukan sebagai pengguna sistem yang Anda konfigurasikan untuk menjalankan komponen ini. Default ke `false`.

`Skipif`

(Opsional) Periksa untuk menentukan apakah akan menjalankan skrip atau tidak. Anda dapat menentukan untuk memeriksa apakah executable berada di jalur atau apakah file ada. Jika output benar, maka Perangkat lunak inti AWS IoT Greengrass melewati langkah itu. Pilih salah satu cek berikut:

- `onpath` *`runnable`* — Periksa apakah runnable berada di jalur sistem. Misalnya, gunakan **`onpath python3`** untuk melewati langkah siklus hidup ini jika Python 3 tersedia.
- `exists` *`file`* — Periksa apakah file ada. Misalnya, gunakan **`exists /tmp/my-configuration.db`** untuk melewati langkah siklus hidup ini jika `/tmp/my-configuration.db` ada.

`Timeout`

(Opsional) Jumlah maksimum waktu dalam detik yang dapat dijalankan skrip sebelum perangkat lunak inti AWS IoT Greengrass mengakhiri proses.

Default: 120 detik.

`Setenv`

(Opsional) Kamus variabel lingkungan untuk menyediakan skrip. Variabel lingkungan ini menimpa variabel yang Anda berikan di `Lifecycle.Setenv`.

`shutdown`

(Opsional) Objek atau string yang mendefinisikan skrip untuk dijalankan ketika komponen dimatikan. Gunakan siklus hidup shutdown untuk mengeksekusi kode yang ingin Anda

jalankan saat komponen dalam status. `STOPPING` Siklus hidup shutdown dapat digunakan untuk menghentikan proses yang dimulai oleh skrip atau. `startup` `run`

Jika Anda memulai proses latar belakang di `startup`, gunakan langkah `shutdown` untuk menghentikan proses itu ketika komponen dimatikan. Misalnya, Anda dapat menentukan langkah `shutdown` yang menghentikan proses MySQL dengan `/etc/init.d/mysqld stop`.

`shutdown`Skrip berjalan setelah komponen memasuki `STOPPING` status. Jika skrip berhasil diselesaikan, komponen memasuki `FINISHED` status.

Objek atau string ini berisi informasi berikut:

`Script`

Skrip yang akan dijalankan.

`RequiresPrivilege`

(Opsional) Anda dapat menjalankan skrip dengan hak istimewa root. Jika Anda menyetel opsi ini`true`, maka perangkat lunak AWS IoT Greengrass Core menjalankan skrip siklus hidup ini sebagai root, bukan sebagai pengguna sistem yang Anda konfigurasikan untuk menjalankan komponen ini. Default ke `false`.

`Skipif`

(Opsional) Periksa untuk menentukan apakah akan menjalankan skrip atau tidak. Anda dapat menentukan untuk memeriksa apakah executable berada di jalur atau apakah file ada. Jika output benar, maka Perangkat lunak inti AWS IoT Greengrass melewati langkah itu. Pilih salah satu cek berikut:

- `onpath` *`runnable`* — Periksa apakah runnable berada di jalur sistem. Misalnya, gunakan **`onpath python3`** untuk melewati langkah siklus hidup ini jika Python 3 tersedia.
- `exists` *`file`* — Periksa apakah file ada. Misalnya, gunakan **`exists /tmp/my-configuration.db`** untuk melewati langkah siklus hidup ini jika `/tmp/my-configuration.db` ada.

`Timeout`

(Opsional) Jumlah waktu maksimum dalam hitungan detik skrip dapat dijalankan sebelum perangkat lunak AWS IoT Greengrass Inti menghentikan proses.

Default: 15 detik.

`Setenv`

(Opsional) Kamus variabel lingkungan untuk menyediakan skrip. Variabel lingkungan ini menimpa variabel yang Anda berikan di `Lifecycle.Setenv`.

`recover`

(Opsional) Objek atau string yang mendefinisikan skrip untuk dijalankan ketika komponen mengalami kesalahan.

Langkah ini berjalan ketika komponen memasuki keadaan `ERRORED`. Jika komponen tersebut memasuki keadaan `ERRORED` tiga kali tanpa berhasil pulih, komponen tersebut akan berubah ke keadaan `BROKEN`. Untuk memperbaiki komponen `BROKEN`, Anda harus men-deploy komponen itu lagi.

Objek atau string ini berisi informasi berikut:

`Script`

Skrip yang akan dijalankan.

`RequiresPrivilege`

(Opsional) Anda dapat menjalankan skrip dengan hak istimewa root. Jika Anda menyetel opsi ini`true`, maka perangkat lunak AWS IoT Greengrass Core menjalankan skrip siklus hidup ini sebagai root, bukan sebagai pengguna sistem yang Anda konfigurasikan untuk menjalankan komponen ini. Default ke `false`.

`Skipif`

(Opsional) Periksa untuk menentukan apakah akan menjalankan skrip atau tidak. Anda dapat menentukan untuk memeriksa apakah executable berada di jalur atau apakah file ada. Jika output benar, maka Perangkat lunak inti AWS IoT Greengrass melewati langkah itu. Pilih salah satu cek berikut:

- `onpath` *`runnable`* — Periksa apakah runnable berada di jalur sistem. Misalnya, gunakan **`onpath python3`** untuk melewati langkah siklus hidup ini jika Python 3 tersedia.
- `exists` *`file`* — Periksa apakah file ada. Misalnya, gunakan **`exists /tmp/my-configuration.db`** untuk melewati langkah siklus hidup ini jika `/tmp/my-configuration.db` ada.

`Timeout`

(Opsional) Jumlah waktu maksimum dalam hitungan detik skrip dapat dijalankan sebelum perangkat lunak AWS IoT Greengrass Inti menghentikan proses.

Default: 60 detik.

`Setenv`

(Opsional) Kamus variabel lingkungan untuk menyediakan skrip. Variabel lingkungan ini menimpa variabel yang Anda berikan di `Lifecycle.Setenv`.

`bootstrap`

(Opsional) Objek atau string yang mendefinisikan skrip yang memerlukan perangkat lunak AWS IoT Greengrass inti atau perangkat inti untuk memulai ulang. Hal ini memungkinkan Anda mengembangkan komponen yang melakukan restart setelah menginstal pembaruan sistem operasi atau pembaruan waktu aktif, misalnya.

> **Note**
>
> Untuk menginstal pembaruan atau dependensi yang tidak memerlukan perangkat lunak atau perangkat AWS IoT Greengrass inti untuk memulai ulang, gunakan siklus hidup penginstalan.

Langkah siklus hidup ini berjalan sebelum langkah siklus hidup penginstalan dalam kasus berikut ketika perangkat lunak AWS IoT Greengrass Core menyebarkan komponen:

- Komponen men-deploy ke perangkat inti untuk pertama kalinya.
- Versi komponen berubah.
- Bootstrap script berubah sebagai hasil dari pembaruan konfigurasi komponen.

Setelah perangkat lunak AWS IoT Greengrass Core menyelesaikan langkah bootstrap untuk semua komponen yang memiliki langkah bootstrap dalam penerapan, perangkat lunak dimulai ulang.

> **Important**
>
> Anda harus mengkonfigurasi perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem untuk memulai ulang perangkat lunak AWS IoT Greengrass Core

atau perangkat inti. Jika Anda tidak mengonfigurasi perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem, perangkat lunak tidak akan dimulai ulang. Untuk informasi selengkapnya, lihat Konfigurasikan inti Greengrass sebagai layanan sistem.

Objek atau string ini berisi informasi berikut:

`BootstrapOnRollback`

> **Note**
>
> Ketika fitur ini diaktifkan, hanya `BootstrapOnRollback` akan berjalan untuk komponen yang telah menyelesaikan atau mencoba menjalankan langkah-langkah siklus hidup bootstrap sebagai bagian dari penerapan target yang gagal. Fitur ini tersedia untuk Greengrass nucleus versi 2.12.0 dan yang lebih baru.

(Opsional) Anda dapat menjalankan langkah-langkah siklus hidup bootstrap sebagai bagian dari penerapan rollback. Jika Anda menyetel opsi ini`true`, langkah-langkah siklus hidup bootstrap yang ditentukan dalam penerapan rollback akan berjalan. Ketika penerapan gagal, versi sebelumnya dari siklus hidup bootstrap komponen akan berjalan lagi selama penerapan rollback.

Default ke `false`.

`Script`

Skrip yang akan dijalankan. Kode keluar dari skrip ini mendefinisikan instruksi restart. Gunakan kode keluar berikut:

- `0`— Jangan me-restart perangkat lunak AWS IoT Greengrass Core atau perangkat inti. Perangkat lunak AWS IoT Greengrass Core masih restart setelah semua komponen bootstrap.
- `100`— Permintaan untuk me-restart perangkat lunak AWS IoT Greengrass Core.
- `101` — Permintaan untuk me-restart perangkat inti.

Kode keluar 100 sampai 199 dicadangkan untuk perilaku khusus. Kode keluar lainnya mewakili kesalahan skrip.

`RequiresPrivilege`

(Opsional) Anda dapat menjalankan skrip dengan hak istimewa root. Jika Anda menyetel opsi ini`true`, maka perangkat lunak AWS IoT Greengrass Core menjalankan skrip siklus hidup ini sebagai root, bukan sebagai pengguna sistem yang Anda konfigurasikan untuk menjalankan komponen ini. Default ke `false`.

`Timeout`

(Opsional) Jumlah maksimum waktu dalam detik yang dapat dijalankan skrip sebelum perangkat lunak inti AWS IoT Greengrass mengakhiri proses.

Default: 120 detik.

`Setenv`

(Opsional) Kamus variabel lingkungan untuk menyediakan skrip. Variabel lingkungan ini menimpa variabel yang Anda berikan di `Lifecycle.Setenv`.

`Selections`

(Opsional) Daftar kunci pilihan yang menentukan bagian Siklus hidup global yang akan dijalankan untuk manifes ini. Dalam siklus hidup global, Anda dapat menentukan langkah-langkah siklus hidup dengan kunci pilihan pada tingkat mana pun untuk memilih sub-bagian dari siklus hidup. Kemudian, perangkat inti menggunakan bagian-bagian yang cocok dengan kunci pilihan dalam manifes ini. Untuk informasi lebih lanjut, lihat bagian contoh siklus hidup global.

> **Important**
>
> Perangkat inti hanya menggunakan siklus hidup global hanya jika manifes ini tidak menentukan siklus hidup.

Anda dapat menentukan kunci pilihan `all` untuk menjalankan bagian dari siklus hidup global yang tidak memiliki kunci pilihan.

`Artifacts`

(Opsional) Daftar objek yang masing-masing menentukan artefak biner untuk komponen pada platform yang ditentukan oleh manifes ini. Misalnya, Anda dapat menentukan kode atau citra sebagai artefak.

Saat komponen digunakan, perangkat lunak AWS IoT Greengrass Core mengunduh artefak ke folder pada perangkat inti. Anda juga dapat menentukan artefak sebagai file arsip yang diekstraki oleh perangkat lunak setelah mengunduhnya.

Anda dapat menggunakan variabel resep untuk mendapatkan jalur ke folder tempat artefak menginstal pada perangkat inti.

- File normal — Gunakan variabel resep artifacts:path untuk mendapatkan jalur ke folder yang berisi artefak. Sebagai contoh, tentukan `{artifacts:path}/my_script.py` dalam resep untuk mendapatkan jalur ke artefak yang memiliki URI `s3://DOC-EXAMPLE-BUCKET/path/to/my_script.py`.
- Arsip yang diekstraksi — Gunakan variabel resep artifacts:decompressedPath untuk mendapatkan jalur ke folder yang berisi artefak arsip yang diekstraksi. Perangkat lunak AWS IoT Greengrass Core mengekstrak setiap arsip ke folder dengan nama yang sama dengan arsip. Sebagai contoh, tentukan `{artifacts:decompressedPath}/my_archive/my_script.py` dalam resep untuk mendapatkan jalur ke artefak arsip `my_script.py` yang memiliki URI `s3://DOC-EXAMPLE-BUCKET/path/to/my_archive.zip`.

> **Note**
>
> Ketika Anda mengembangkan komponen dengan artefak arsip pada perangkat inti lokal, Anda mungkin tidak memiliki URI untuk artefak itu. Untuk menguji komponen Anda dengan `Unarchive` yang mengekstraksi artefak, tentukan URI tempat nama file cocok dengan nama file artefak arsip Anda. Anda dapat menentukan URI di mana Anda mengharapkan untuk mengunggah artefak arsip, atau Anda dapat menentukan URI placeholder yang baru. Misalnya, untuk mengekstraksi artefak `my_archive.zip` selama deployment lokal, Anda dapat menentukan `s3://DOC-EXAMPLE-BUCKET/my_archive.zip`.

Setiap objek berisi informasi berikut:

`URI`

URI artefak dalam bucket S3. Perangkat lunak AWS IoT Greengrass Core mengambil artefak dari URI ini saat komponen diinstal, kecuali artefak sudah ada di perangkat. Setiap artefak harus memiliki nama file yang unik dalam setiap manifes.

`Unarchive`

(Opsional) Jenis arsip yang akan dibongkar. Pilih dari salah satu pilihan berikut:

- `NONE` – File ini bukanlah arsip yang akan dibongkar. Perangkat lunak inti AWS IoT Greengrass menginstal artefak ke folder pada perangkat inti. Anda dapat menggunakan variabel resep artifacts:path untuk mendapatkan jalur ke folder ini.
- `ZIP` – File ini adalah arsip ZIP. Perangkat lunak AWS IoT Greengrass Core mengekstrak arsip ke folder dengan nama yang sama dengan arsip. Anda dapat menggunakan variabel resep artifacts:decompressedPath untuk mendapatkan jalur ke folder ini.

Default ke `NONE`.

`Permission`

(Opsional) Sebuah objek yang mendefinisikan izin akses yang akan ditentukan untuk file artefak ini. Anda dapat mengatur izin membaca dan izin mengeksekusi.

> **Note**
>
> Anda tidak dapat mengatur izin tulis, karena perangkat lunak AWS IoT Greengrass Core tidak mengizinkan komponen untuk mengedit file artefak di folder artefak. Untuk mengedit file artefak dalam komponen, salin ke lokasi lain atau publikasikan dan deploy file artefak yang baru.

Jika Anda mendefinisikan artefak sebagai arsip untuk dibongkar, maka perangkat lunak AWS IoT Greengrass Core menetapkan izin akses ini pada file yang dibongkar dari arsip. Perangkat lunak AWS IoT Greengrass Core menetapkan izin akses folder `ALL` untuk `Read` dan`Execute`. Hal ini memungkinkan komponen untuk melihat file yang dibongkar dalam folder. Untuk mengatur izin pada masing-masing file dari arsip, Anda dapat mengatur izin di skrip install lifecycle.

Objek ini berisi informasi berikut:

`Read`

(Opsional) Izin baca yang akan ditetapkan untuk file artefak ini. Untuk mengizinkan komponen lain untuk mengakses artefak ini, seperti komponen yang bergantung pada komponen ini, tentukan `ALL`. Pilih dari salah satu pilihan berikut:

- `NONE` – File ini tidak dapat dibaca.
- `OWNER` – File ini dapat dibaca oleh pengguna sistem yang Anda konfigurasikan untuk menjalankan komponen ini.

- `ALL` – File ini dapat dibaca oleh semua pengguna.

Default ke `OWNER`.

`Execute`

(Opsional) Izin run yang akan ditetapkan untuk file artefak ini. Izin `Execute` menyiratkan izin `Read`. Misalnya, jika Anda menentukan `ALL` untuk `Execute`, maka semua pengguna dapat membaca dan menjalankan file artefak ini.

Pilih dari salah satu pilihan berikut:

- `NONE` – File ini tidak dapat dijalankan.
- `OWNER` – File ini dapat dibaca oleh pengguna sistem yang Anda konfigurasikan untuk menjalankan komponen ini.
- `ALL` – File ini dapat dijalankan oleh semua pengguna.

Default ke `NONE`.

`Digest`

(Hanya baca) Cryptographic digest hash dari artefak. Saat Anda membuat komponen, AWS IoT Greengrass gunakan algoritma hash untuk menghitung hash dari file artefak. Kemudian, ketika Anda men-deploy komponen, inti Greengrass menghitung hash dari artefak terunduh dan membandingkan hash dengan digest ini untuk memverifikasi artefak sebelum instalasi. Jika hash tidak cocok dengan digest, deployment gagal.

Jika Anda menyetel parameter ini, AWS IoT Greengrass ganti nilai yang Anda tetapkan saat membuat komponen.

`Algorithm`

(Read-only) Algoritma hash yang AWS IoT Greengrass digunakan untuk menghitung hash intisari artefak.

Jika Anda menyetel parameter ini, AWS IoT Greengrass ganti nilai yang Anda tetapkan saat membuat komponen.

`Lifecycle`

Objek yang mendefinisikan cara menginstal dan menjalankan komponen. Perangkat inti hanya menggunakan siklus hidup global hanya jika manifes yang akan digunakan tidak menentukan siklus hidup.

> **Note**
>
> Anda mendefinisikan siklus hidup ini di luar manifes. Anda juga dapat menentukan manifest siklus hidup yang berlaku untuk platform yang sesuai dengan manifes itu.

Dalam siklus hidup global, Anda dapat menentukan siklus hidup yang berjalan untuk kunci pilihan tertentu yang Anda tentukan di setiap manifes. Kunci pilihan adalah string yang mengidentifikasi bagian dari siklus hidup global yang akan dijalankan untuk setiap manifes.

Kunci pilihan `all` adalah default pada setiap bagian tanpa kunci seleksi. Artinya Anda dapat menggunakan pilihan `all` dalam manifes untuk menjalankan bagian siklus hidup global tanpa kunci seleksi. Anda tidak perlu menentukan opsi kunci pilihan `all` dalam siklus hidup global.

Jika manifes tidak menentukan siklus hidup atau kunci pilihan, default perangkat inti akan menggunakan pilihan `all`. Ini berarti bahwa dalam kasus ini, perangkat inti menggunakan bagian siklus hidup global yang tidak menggunakan kunci pilihan.

Objek ini berisi informasi yang sama seperti manifes siklus hidup, namun Anda dapat menentukan kunci pilihan pada tingkat mana pun untuk memilih sub-bagian dari siklus hidup.

> **Tip**
>
> Kami menyarankan Anda menggunakan hanya huruf kecil untuk setiap kunci pilihan untuk menghindari pertentangan antara kunci pilihan dan kunci siklus hidup. Kunci siklus hidup dimulai dengan huruf kapital.

Example Contoh siklus hidup global dengan kunci pilihan tingkat atas

```
Lifecycle:
  key1:
    install:
      Skipif: either onpath executable or exists file
      Script: command1
  key2:
    install:
      Script: command2
  all:
    install:
```

```
      Script: command3
```

Example Contoh siklus hidup global dengan kunci pilihan tingkat bawah

```
Lifecycle:
  install:
    Script:
      key1: command1
      key2: command2
      all: command3
```

Example Contoh siklus hidup global dengan berbagai tingkatan kunci pilihan

```
Lifecycle:
  key1:
    install:
      Skipif: either onpath executable or exists file
      Script: command1
  key2:
    install:
      Script: command2
  all:
    install:
      Script:
        key3: command3
        key4: command4
        all: command5
```

Variabel resep

Variabel resep mengekspos informasi dari komponen saat ini dan inti yang dapat Anda gunakan dalam resep Anda. Misalnya, Anda dapat menggunakan variabel resep untuk meneruskan parameter konfigurasi komponen ke aplikasi yang Anda jalankan dalam skrip siklus hidup.

Anda dapat menggunakan variabel resep di bagian resep komponen berikut:

- Definisi siklus hidup.
- Definisi konfigurasi komponen, jika Anda menggunakan Greengrass nucleus v2.6.0 atau yang lebih baru dan mengatur opsi konfigurasi ke. interpolateComponentConfiguration`true` Anda juga dapat menggunakan variabel resep saat menerapkan pembaruan konfigurasi komponen.

Variabel resep menggunakan sintaks `{recipe_variable}`. Kawat lengkung menunjukkan variabel resep.

AWS IoT Greengrass mendukung variabel resep berikut:

`component_dependency_name``:configuration:`*`json_pointer`*

Nilai parameter konfigurasi untuk komponen yang didefinisikan oleh resep ini atau untuk komponen yang komponen ini tergantung padanya.

Anda dapat menggunakan variabel ini untuk memberikan parameter pada skrip yang Anda jalankan dalam siklus hidup komponen.

> **Note**
>
> AWS IoT Greengrass mendukung variabel resep ini hanya dalam definisi siklus hidup komponen.

Variabel resep ini memiliki masukan berikut:

- `component_dependency_name` — (Opsional) Nama dependensi komponen pada kueri. Hilangkan segmen ini untuk melakukan kueri atas komponen yang menentukan resep ini. Anda hanya dapat menentukan dependensi langsung.
- `json_pointer` - Pointer JSON pada nilai konfigurasi yang akan dievaluasi. JSON pointer dimulai dengan garis miring `/`. Untuk mengidentifikasi nilai dalam konfigurasi komponen bersusun, gunakan garis miring ke depan (`/`) untuk memisahkan kunci-kunci untuk setiap tingkat dalam konfigurasi. Anda dapat menggunakan nomor sebagai kunci untuk menentukan indeks dalam daftar. Untuk informasi selengkapnya, lihat spesifikasi pointer JSON.

  AWS IoT Greengrass Core menggunakan pointer JSON untuk resep dalam format YAMG.

Pointer JSON dapat mereferensikan jenis node berikut:

- Sebuah simpul nilai. AWS IoT Greengrass Core menggantikan variabel resep dengan representasi string dari nilai. Nilai kosong berubah ke `null` sebagai string.
- Sebuah simpul objek. AWS IoT Greengrass Core menggantikan variabel resep dengan representasi string JSON serial dari objek itu.
- Tidak ada simpul. AWS IoT Greengrass Core tidak menggantikan variabel resep.

Misalnya, variabel resep `{configuration:/Message}` mengambil nilai kunci `Message` dalam konfigurasi komponen tersebut. Variabel resep `{com.example.MyComponentDependency:configuration:/server/port}` mengambil nilai `port` di konfigurasi objek `server` dari dependensi komponen.

component_dependency_name`:artifacts:path`

Jalur akar artefak untuk komponen yang didefinisikan oleh resep ini atau untuk komponen yang komponen ini tergantung padanya.

Saat komponen diinstal, salin AWS IoT Greengrass artefak komponen ke folder yang diekspos variabel ini. Anda dapat menggunakan variabel ini untuk mengidentifikasi lokasi skrip yang akan dijalankan dalam siklus hidup komponen, misalnya.

Folder di jalur ini bersifat hanya-baca. Untuk mengubah file artefak, salin file ke lokasi lain, seperti direktori kerja saat ini (`$PWD` atau `.`). Kemudian, ubah file di sana.

Untuk membaca atau menjalankan artefak dari dependensi komponen, izin `Read` atau `Execute` artefak itu harus `ALL`. Untuk informasi lebih lanjut, lihat izin artefak yang Anda tentukan dalam resep komponen.

Variabel resep ini memiliki masukan berikut:

- `component_dependency_name` — (Opsional) Nama dependensi komponen pada kueri. Hilangkan segmen ini untuk melakukan kueri atas komponen yang menentukan resep ini. Anda hanya dapat menentukan dependensi langsung.

component_dependency_name`:artifacts:decompressedPath`

Jalur akar artefak dari artefak arsip yang terdekompresi untuk komponen yang didefinisikan oleh resep ini atau untuk komponen yang komponen ini tergantung padanya.

Saat komponen diinstal, AWS IoT Greengrass buka artefak arsip komponen ke folder yang diekspos variabel ini. Anda dapat menggunakan variabel ini untuk mengidentifikasi lokasi skrip yang akan dijalankan dalam siklus hidup komponen, misalnya.

Setiap artefak di-unzip ke folder dalam jalur dekompresi, di mana folder itu memiliki nama yang sama dengan artefak dikurangi ekstensinya. Sebagai contoh, sebuah artefak ZIP bernama `models.zip` dibongkar ke folder `{artifacts:decompressedPath}/models`.

Folder di jalur ini bersifat hanya-baca. Untuk mengubah file artefak, salin file ke lokasi lain, seperti direktori kerja saat ini (`$PWD` atau `.`). Kemudian, ubah file di sana.

Untuk membaca atau menjalankan artefak dari dependensi komponen, izin `Read` atau `Execute` artefak itu harus `ALL`. Untuk informasi lebih lanjut, lihat izin artefak yang Anda tentukan dalam resep komponen.

Variabel resep ini memiliki masukan berikut:

- `component_dependency_name` — (Opsional) Nama dependensi komponen pada kueri. Hilangkan segmen ini untuk melakukan kueri atas komponen yang menentukan resep ini. Anda hanya dapat menentukan dependensi langsung.

`component_dependency_name``:work:path`

Fitur ini tersedia untuk v2.0.4 dan versi yang lebih baru dari komponen inti Greengrass.

Jalur akar artefak untuk komponen yang didefinisikan oleh resep ini atau untuk komponen yang komponen ini tergantung padanya. Nilai variabel resep ini setara dengan output dari variabel lingkungan `$PWD` dan perintah pwd ketika dijalankan dari konteks komponen.

Anda dapat menggunakan variabel resep ini untuk berbagi file antara komponen dan dependensi.

Folder di jalur ini dapat dibaca dan ditulis oleh komponen yang didefinisikan oleh resep ini dan oleh komponen lain yang dijalankan sebagai pengguna dan kelompok yang sama.

Variabel resep ini memiliki masukan berikut:

- `component_dependency_name` — (Opsional) Nama dependensi komponen pada kueri. Hilangkan segmen ini untuk melakukan kueri atas komponen yang menentukan resep ini. Anda hanya dapat menentukan dependensi langsung.

`kernel:rootPath`

Jalur akar AWS IoT Greengrass inti.

`iot:thingName`

Fitur ini tersedia untuk v2.3.0 dan versi kemudian dari komponen inti Greengrass.

Nama AWS IoT benda perangkat inti.

Contoh resep

Anda dapat mereferensikan contoh resep berikut untuk membantu Anda membuat resep untuk komponen Anda.

AWS IoT Greengrass mengkurasi indeks komponen Greengrass, yang disebut Greengrass Software Catalog. Katalog ini melacak komponen Greengrass yang dikembangkan oleh komunitas Greengrass. Dari katalog ini, Anda dapat mengunduh, memodifikasi, dan menyebarkan komponen untuk membuat aplikasi Greengrass Anda. Untuk informasi selengkapnya, lihat Komponen komunitas.

Topik

- Resep komponen Hello World
- Contoh komponen waktu aktif Python
- Komponen resep yang menentukan beberapa kolom

Resep komponen Hello World

Resep berikut menjelaskan komponen Hello World yang menjalankan skrip Python. Komponen ini mendukung semua platform dan menerima `Message` parameter yang AWS IoT Greengrass diteruskan sebagai argumen ke skrip Python. Ini adalah resep untuk komponen Hello World di Memulai tutorial.

JSON

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

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

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.HelloWorld
ComponentVersion: '1.0.0'
ComponentDescription: My first AWS IoT Greengrass component.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    Message: world
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      run: |
        python3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
  - Platform:
      os: windows
    Lifecycle:
      run: |
        py -3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
```

Contoh komponen waktu aktif Python

Resep berikut menjelaskan komponen yang menginstal Python. Komponen ini mendukung perangkat Linux 64-bit.

JSON

```
{
```

```
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PythonRuntime",
  "ComponentDescription": "Installs Python 3.7",
  "ComponentPublisher": "Amazon",
  "ComponentVersion": "3.7.0",
  "Manifests": [
    {
      "Platform": {
        "os": "linux",
        "architecture": "amd64"
      },
      "Lifecycle": {
        "install": "apt-get update\napt-get install python3.7"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PythonRuntime
ComponentDescription: Installs Python 3.7
ComponentPublisher: Amazon
ComponentVersion: '3.7.0'
Manifests:
  - Platform:
      os: linux
      architecture: amd64
    Lifecycle:
      install: |
        apt-get update
        apt-get install python3.7
```

Komponen resep yang menentukan beberapa kolom

Resep komponen berikut menggunakan beberapa kolom resep.

JSON

```
{
```

```
"RecipeFormatVersion": "2020-01-25",
"ComponentName": "com.example.FooService",
"ComponentDescription": "Complete recipe for AWS IoT Greengrass components",
"ComponentPublisher": "Amazon",
"ComponentVersion": "1.0.0",
"ComponentConfiguration": {
  "DefaultConfiguration": {
    "TestParam": "TestValue"
  }
},
"ComponentDependencies": {
  "BarService": {
    "VersionRequirement": "^1.1.0",
    "DependencyType": "SOFT"
  },
  "BazService": {
    "VersionRequirement": "^2.0.0"
  }
},
"Manifests": [
  {
    "Platform": {
      "os": "linux",
      "architecture": "amd64"
    },
    "Lifecycle": {
      "install": {
        "Skipif": "onpath git",
        "Script": "sudo apt-get install git"
      },
      "Setenv": {
        "environment_variable1": "variable_value1",
        "environment_variable2": "variable_value2"
      }
    },
    "Artifacts": [
      {
        "URI": "s3://DOC-EXAMPLE-BUCKET/hello_world.zip",
        "Unarchive": "ZIP"
      },
      {
        "URI": "s3://DOC-EXAMPLE-BUCKET/hello_world_linux.py"
      }
    ]
```

```
    },
    {
      "Lifecycle": {
        "install": {
          "Skipif": "onpath git",
          "Script": "sudo apt-get install git",
          "RequiresPrivilege": "true"
        }
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/hello_world.py"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.FooService
ComponentDescription: Complete recipe for AWS IoT Greengrass components
ComponentPublisher: Amazon
ComponentVersion: 1.0.0
ComponentConfiguration:
  DefaultConfiguration:
    TestParam: TestValue
ComponentDependencies:
  BarService:
    VersionRequirement: ^1.1.0
    DependencyType: SOFT
  BazService:
    VersionRequirement: ^2.0.0
Manifests:
  - Platform:
      os: linux
      architecture: amd64
    Lifecycle:
      install:
        Skipif: onpath git
        Script: sudo apt-get install git
```

```
    Setenv:
      environment_variable1: variable_value1
      environment_variable2: variable_value2
  Artifacts:
    - URI: 's3://DOC-EXAMPLE-BUCKET/hello_world.zip'
      Unarchive: ZIP
    - URI: 's3://DOC-EXAMPLE-BUCKET/hello_world_linux.py'
- Lifecycle:
    install:
      Skipif: onpath git
      Script: sudo apt-get install git
      RequiresPrivilege: 'true'
  Artifacts:
    - URI: 's3://DOC-EXAMPLE-BUCKET/hello_world.py'
```

Referensi variabel lingkungan komponen

Perangkat lunak inti AWS IoT Greengrass menetapkan variabel lingkungan ketika menjalankan skrip siklus hidup untuk komponen. Anda bisa mendapatkan variabel lingkungan ini dalam komponen Anda untuk mendapatkan nama objek, Wilayah AWS, dan versi nukleus Greengrass. Perangkat lunak ini juga menetapkan variabel lingkungan yang diperlukan komponen Anda untuk menggunakan SDK komunikasi antar proses dan untuk berinteraksi dengan layanan AWS.

Anda juga dapat mengatur variabel lingkungan kustom untuk skrip siklus hidup komponen Anda. Untuk informasi lebih lanjut, lihat Setenv

Perangkat lunak inti AWS IoT Greengrass menetapkan variabel lingkungan berikut:

`AWS_IOT_THING_NAME`

Nama objek AWS IoT yang mewakili perangkat inti Greengrass ini.

`AWS_REGION`

Wilayah AWS di mana perangkat inti Greengrass ini beroperasi.

SDK AWS menggunakan variabel lingkungan ini untuk mengidentifikasi Wilayah default yang akan digunakan. Variabel ini setara dengan `AWS_DEFAULT_REGION`.

`AWS_DEFAULT_REGION`

Wilayah AWS di mana perangkat inti Greengrass ini beroperasi.

AWS CLI menggunakan variabel lingkungan ini untuk mengidentifikasi Wilayah default yang akan digunakan. Variabel ini setara dengan `AWS_REGION`.

`GGC_VERSION`

Versi komponen inti Greengrass yang berjalan pada perangkat inti Greengrass ini.

`GG_ROOT_CA_PATH`

Fitur ini tersedia untuk v2.5.5 dan versi kemudian dari komponen inti Greengrass.

Jalur sertifikat (CA) akar yang digunakan inti Greengrass.

`AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT`

Path ke soket IPC yang digunakan oleh komponen untuk berkomunikasi dengan perangkat lunak inti AWS IoT Greengrass. Untuk informasi selengkapnya, lihat Gunakan AWS IoT Device SDK untuk berkomunikasi dengan inti Greengrass, komponen lain, dan AWS IoT Core.

`SVCUID`

Token rahasia yang digunakan oleh komponen untuk terhubung ke soket IPC dan berkomunikasi dengan perangkat lunak inti AWS IoT Greengrass. Untuk informasi selengkapnya, lihat Gunakan AWS IoT Device SDK untuk berkomunikasi dengan inti Greengrass, komponen lain, dan AWS IoT Core.

`AWS_CONTAINER_AUTHORIZATION_TOKEN`

Token rahasia yang digunakan komponen untuk mengambil kredensial dari komponen layanan pertukaran token.

`AWS_CONTAINER_CREDENTIALS_FULL_URI`

Token rahasia yang diminta komponen untuk mengambil kredensial dari komponen layanan pertukaran token.

Deploy komponen AWS IoT Greengrass ke perangkat

Anda dapat menggunakan AWS IoT Greengrass untuk menyebarkan komponen ke perangkat atau grup perangkat. Anda menggunakan penerapan untuk menentukan komponen dan konfigurasi yang dikirim ke perangkat. AWS IoT Greengrassmenyebarkan ke target, AWS IoT benda atau kelompok hal yang mewakili perangkat inti Greengrass. AWS IoT Greengrassmenggunakan AWS IoT Corepekerjaan untuk menyebarkan ke perangkat inti Anda. Anda dapat mengonfigurasi bagaimana pekerjaan diluncurkan ke perangkat Anda.

Penerapan perangkat inti

Setiap perangkat inti menjalankan komponen penerapan untuk perangkat itu. Penerapan baru ke target yang sama menimpa penerapan sebelumnya ke target. Saat Anda membuat penerapan, Anda menentukan komponen dan konfigurasi yang akan diterapkan ke perangkat lunak inti perangkat yang ada.

Ketika Anda merevisi deployment untuk target, Anda mengganti komponen dari revisi sebelumnya dengan komponen dalam revisi baru. Misalnya, Anda menyebarkan Secrets manager komponen Manajer log dan ke grup `TestGroup` benda. Kemudian Anda membuat penerapan lain untuk `TestGroup` yang hanya menentukan komponen manajer rahasia. Akibatnya, perangkat inti dalam grup itu tidak lagi menjalankan pengelola log.

Resolusi ketergantungan platform

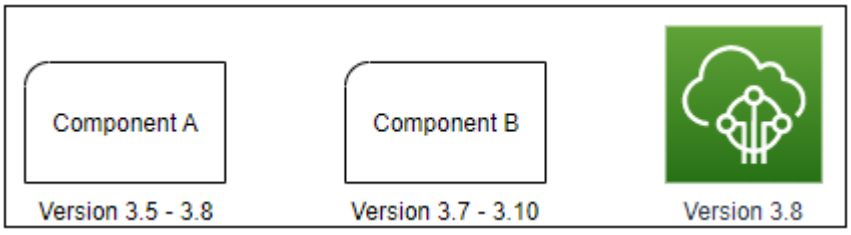
Ketika perangkat inti menerima penerapan, ia memeriksa untuk memastikan bahwa komponen tersebut kompatibel dengan perangkat inti. Misalnya, jika Anda menyebarkan Firehose ke target Windows, penerapan akan gagal.

Resolusi ketergantungan komponen

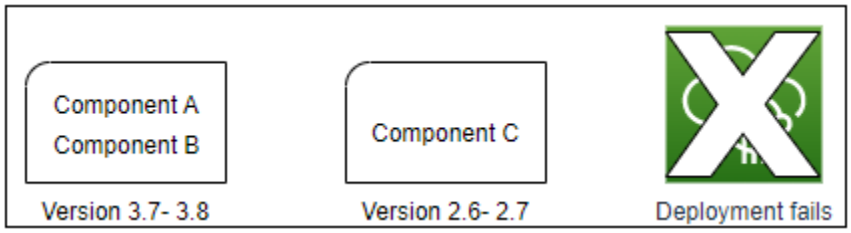
Perangkat inti juga memeriksa apakah setiap dependensi komponen kompatibel dengan batasan versi untuk penerapan komponen lain ke grup hal ini. Jika batasan versi untuk komponen tumpang tindih, Greengrass menggunakan versi komponen tertinggi yang berlaku. Sebagai contoh:

- Anda menerapkan komponen A ke`TestGroup`. Komponen A tergantung pada `com.example.PythonRuntime` versi komponen 3.5 - 3.10.
- Anda kemudian menerapkan komponen B ke`TestGroup`. Komponen B tergantung pada `com.example.PythonRuntime` versi komponen 3.7 hingga 3.8.

Akibatnya, perangkat inti `TestGroup` menentukan bahwa mereka dapat menerapkan versi 3.8 `com.example.PythonRuntime` komponen karena versi ini adalah versi tertinggi yang berlaku di mana batasan versi tumpang tindih.



Anda kemudian menerapkan komponen C ke`TestGroup`. Komponen C tergantung pada `com.example.PythonRuntime` versi komponen 2.6 - 2.7. Penerapan ini gagal karena tidak ada versi komponen yang memenuhi batasan 2.6 - 2.7 dan 3.7 - 3.8.



Menghapus perangkat dari grup benda

Saat Anda menghapus perangkat inti dari grup sesuatu, perilaku penerapan komponen bergantung pada versi inti Greengrass yang dijalankan perangkat inti.

2.5.1 and later

Saat Anda menghapus perangkat inti dari grup sesuatu, perilakunya bergantung pada apakah AWS IoT kebijakan memberikan `greengrass:ListThingGroupsForCoreDevice` izin tersebut. Untuk informasi selengkapnya tentang izin dan AWS IoT kebijakan untuk perangkat inti ini, lihatAutentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.

- Jika AWS IoT kebijakan memberikan izin ini

  Saat Anda menghapus perangkat inti dari grup benda, AWS IoT Greengrass hapus komponen grup benda saat penerapan dilakukan ke perangkat berikutnya. Jika komponen pada perangkat disertakan dalam penerapan berikutnya, komponen tersebut tidak dihapus dari perangkat.

- Jika AWS IoT kebijakan tidak memberikan izin ini

  Saat Anda menghapus perangkat inti dari grup objek, AWS IoT Greengrass tidak menghapus komponen grup objek itu dari perangkat.

  Untuk menghapus komponen dari perangkat, gunakan perintah deployment create dari CLI Greengrass. Tentukan komponen yang akan dihapus dengan argumen `--remove`, dan tentukan grup objek dengan argumen `--groupId`.

2.5.0

Saat Anda menghapus perangkat inti dari grup benda, AWS IoT Greengrass hapus komponen grup benda saat penerapan dilakukan ke perangkat berikutnya. Jika komponen pada perangkat disertakan dalam penerapan berikutnya, komponen tersebut tidak dihapus dari perangkat.

Perilaku ini mengharuskan AWS IoT kebijakan perangkat inti memberikan `greengrass:ListThingGroupsForCoreDevice` izin. Jika perangkat inti tidak memiliki izin ini, perangkat inti gagal menerapkan penerapan. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi perangkat untuk AWS IoT Greengrass.

2.0.x - 2.4.x

Saat Anda menghapus perangkat inti dari grup objek, AWS IoT Greengrass tidak menghapus komponen grup objek itu dari perangkat.

Untuk menghapus komponen dari perangkat, gunakan perintah deployment create dari CLI Greengrass. Tentukan komponen yang akan dihapus dengan argumen `--remove`, dan tentukan grup objek dengan argumen `--groupId`.

Deployment

Deployment bersifat terus menerus. Saat Anda membuat deployment, AWS IoT Greengrass meluncurkan deployment untuk menargetkan perangkat yang sedang online. Jika perangkat target tidak online, maka perangkat akan menerima deployment saat berikutnya ketika terhubung ke AWS IoT Greengrass. Saat Anda menambahkan perangkat inti ke grup objek target, AWS IoT Greengrassmengirimkan perangkat deployment terbaru untuk grup objek tersebut.

Sebelum perangkat inti menyebarkan komponen, secara default akan memberi tahu setiap komponen pada perangkat. Komponen Greengrass dapat menanggapi pemberitahuan untuk menunda penerapan. Anda mungkin ingin menunda penerapan jika perangkat memiliki tingkat baterai rendah atau menjalankan proses yang tidak dapat diganggu. Untuk informasi selengkapnya, lihat Tutorial: Mengembangkan komponen Greengrass yang menunda pembaruan komponen. Saat Anda membuat penerapan, Anda dapat mengonfigurasinya untuk diterapkan tanpa memberi tahu komponen.

Setiap obje target atau grup objek dapat memiliki satu deployment pada satu waktu. Ini berarti bahwa ketika Anda membuat deployment untuk suatu target, AWS IoT Greengrass tidak lagi men-deploy revisi sebelumnya dari deployment target tersebut.

Opsi deployment

Deployment menyediakan beberapa opsi yang memungkinkan Anda mengontrol perangkat mana yang menerima deployment dan cara pembaruan itu men-deploy. Saat Anda membuat deployment, Anda dapat mengonfigurasi opsi berikut:

- AWS IoT Greengrasskomponen

  Tentukan komponen yang akan diinstal dan dijalankan pada perangkat target. Komponen AWS IoT Greengrass adalah modul perangkat lunak yang Anda deploy dan berjalan pada perangkat inti Greengrass. Perangkat hanya menerima komponen jika komponen mendukung platform perangkat. Hal ini memungkinkan Anda men-deploy ke grup perangkat bahkan jika perangkat target itu berjalan di beberapa platform. Jika komponen tidak mendukung platform perangkat, komponen tidak di-deploy ke perangkat.

  Anda dapat men-deploy komponen kustom dan komponen yang disediakan oleh AWS untuk perangkat Anda. Saat Anda menggunakan suatu komponen, AWS IoT Greengrass mengidentifikasi dependensi komponen dan men-deploy-nya juga. Lihat informasi yang lebih lengkap di Kembangkan AWS IoT Greengrass komponen dan Komponen yang disediakan oleh AWS.

  Anda menentukan versi dan konfigurasi update untuk yag akan di-deploy untuk untuk setiap komponen. Pembaruan konfigurasi menentukan cara mengubah konfigurasi komponen yang ada pada perangkat inti, atau konfigurasi default komponen jika komponen itiu tidak ada pada perangkat inti. Anda dapat menentukan nilai konfigurasi yang akan di-reset ke nilai default dan nilai-nilai konfigurasi baru yang akan digabungkan ke perangkat inti. Saat perangkat inti menerima penerapan untuk target yang berbeda, dan setiap penerapan menentukan versi komponen yang kompatibel, perangkat inti menerapkan pembaruan konfigurasi secara berurutan berdasarkan stempel waktu saat Anda membuat penerapan. Untuk informasi selengkapnya, lihat Perbarui konfigurasi komponen.

  > **Important**
  >
  > Saat Anda menerapkan komponen, AWS IoT Greengrass instal versi terbaru yang didukung dari semua dependensi komponen tersebut. Karena ini, versi patch baru dari komponen publik yang disediakan oleh AWS mungkin secara otomatis di-deploy ke perangkat inti Anda jika Anda menambahkan perangkat baru ke grup objek, atau Anda memperbarui deployment yang menargetkan perangkat tersebut. Beberapa pembaruan

otomatis, seperti pembaruan inti, dapat menyebabkan perangkat Anda memulai ulang secara tiba-tiba.
Untuk mencegah pembaruan yang tidak diinginkan untuk komponen yang berjalan di perangkat Anda, sebaiknya sertakan versi komponen yang Anda inginkan secara langsung saat membuat deployment. Untuk informasi selengkapnya tentang perilaku pembaruan untuk perangkat lunak Inti AWS IoT Greengrass, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

- Kebijakan penyebaran

  Tentukan kapan waktu yang aman untuk menggunakan konfigurasi dan apa yang harus dilakukan jika deployment gagal. Anda dapat menentukan apakah akan menunggu atau tidak komponen melaporkan bahwa ia dapat memperbarui. Anda juga dapat menentukan apakah akan memutar kembali perangkat ke konfigurasi sebelumnya atau tidak jika ia menerapkan deployment yang gagal.

- Hentikan konfigurasi

  Tentukan kapan dan bagaimana menghentikan deployment. Deployment berhenti dan gagal jika kriteria yang Anda tetapkan terpenuhi. Sebagai contoh, Anda dapat mengonfigurasi deployment untuk berhenti jika persentase perangkat gagal untuk menerapkan deployment itu setelah jumlah minimum perangkat menerimanya.

- Konfigurasi peluncuran

  Tentukan tingkat di mana deployment meluncurkan ke perangkat target. Anda dapat mengonfigurasi kenaikan tingkat eksponensial dengan batas tingkat minimum dan maksimum.

- Konfigurasi batas waktu

  Tentukan jumlah maksimum waktu setiap perangkat harus menerapkan deployment. Jika perangkat melebihi durasi yang Anda tentukan, maka perangkat akan gagal menerapkan deployment.

Important

Komponen khusus dapat menentukan artefak dalam bucket S3. Saat perangkat lunak inti AWS IoT Greengrass men-deploy komponen, ia mengunduh artefak komponen dari AWS Cloud. Peran perangkat inti tidak mengizinkan akses ke bucket S3 secara default. Untuk men-deploy komponen kustom yang menentukan artefak dalam bucket S3, peran perangkat

inti harus memberikan izin untuk men-download artefak dari bucket tersebut. Lihat informasi yang lebih lengkap di Izinkan akses ke bucket S3 untuk artefak komponen.

Topik

Buat deployment

Anda dapat membuat deployment yang menargetkan objek atau grup objek.

Ketika Anda membuat deployment, Anda mengonfigurasi komponen perangkat lunak yang akan di-deploy dan bagaimana tugas deployment keluar ke target perangkat. Anda dapat menentukan deployment dalam file JSON yang Anda berikan ke AWS CLI.

Target deployment menentukan perangkat yang Anda inginkan untuk menjalankan komponen Anda. Untuk men-deploy ke satu perangkat inti, tentukan objek. Untuk men-deploy ke beberapa perangkat inti, tentukan grup objek yang mencakup perangkat tersebut. Untuk informasi lebih lanjut tentang cara mengonfigurasi grup objek, lihat Grup objek statis dan Grup objek dinamis di Panduan Developer AWS IoT.

Ikuti langkah-langkah di bagian ini untuk membuat penyebaran ke target. Untuk informasi selengkapnya tentang cara memperbarui komponen perangkat lunak pada target yang memiliki deployment, lihat Revisi deployment.

⚠ Warning

CreateDeploymentOperasi dapat menghapus komponen dari perangkat inti. Jika komponen ada di penerapan sebelumnya dan bukan penerapan baru, perangkat inti akan menghapus pemasangan komponen tersebut. Untuk menghindari mencopot pemasangan komponen, pertama-tama gunakan ListDeploymentsoperasi untuk memeriksa apakah target penerapan sudah memiliki penerapan yang ada. Kemudian, gunakan GetDeploymentoperasi untuk memulai dari penerapan yang ada saat Anda membuat penerapan baru.

Untuk membuat deployment (AWS CLI)

1. Buat file bernama `deployment.json`, lalu salin objek JSON berikut ke dalam file. Ganti *`targetArn`* dengan ARN objek AWS IoT atau grup objek yang ditargetkan untuk deployment. ARN objek dan grup objek memiliki format berikut:

   - Objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
   - Grup objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*

   ```
   {
     "targetArn": "targetArn"
   }
   ```

2. Periksa apakah target penerapan memiliki penerapan yang sudah ada yang ingin Anda revisi. Lakukan hal-hal berikut:

   a. Jalankan perintah berikut untuk membuat daftar penerapan untuk target penyebaran. Ganti *`targetArn`* dengan ARN target objek AWS IoT atau grup objek.

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   Tanggapan berisi daftar dengan deployment terbaru untuk target. Jika responsnya kosong, target tidak memiliki penerapan yang ada, dan Anda dapat melompat keStep 3. Jika tidak, salin `deploymentId` dari respons untuk digunakan pada langkah berikutnya.

   > **Note**
   >
   > Anda juga dapat merevisi deployment selain revisi terbaru untuk target. Tentukan `--history-filter ALL` untuk mencantumkan semua deployment untuk target. Kemudian, salin ID penerapan yang ingin Anda revisi.

   b. Jalankan perintah berikut untuk mendapatkan detail penerapan. Detail ini mencakup metadata, komponen, dan konfigurasi pekerjaan. Ganti *`deploymentId`* dengan ID dari langkah sebelumnya.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   Tanggapan berisi detail deployment.

c. Salin salah satu pasangan kunci-nilai berikut dari respons perintah sebelumnya ke dalam. `deployment.json` Anda dapat mengubah nilai-nilai ini untuk penerapan baru.

- `deploymentName`— Nama penyebaran.
- `components`— Komponen penyebaran. Untuk menghapus komponen, hapus dari objek ini.
- `deploymentPolicies`— Kebijakan penyebaran.
- `iotJobConfiguration`— Konfigurasi pekerjaan penerapan.
- `tags`— Tag penyebaran.

3. (Opsional) Tentukan nama untuk penerapan. Ganti *`DeploymentName`* dengan nama penyebaran.

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName"
}
```

4. Tambahkan setiap komponen untuk men-deploy perangkat target. Untuk melakukannya, tambahkan pasangan nilai kunci ke objek `components`, di mana kunci adalah nama komponen, dan nilai adalah objek yang berisi detail untuk komponen tersebut. Tentukan detail berikut untuk setiap komponen yang Anda tambahkan:

- `version` — Versi komponen yang akan di-deploy.
- `configurationUpdate` — Pembaruan konfigurasi yang akan di-deploy. Pembaruan adalah operasi patch yang memodifikasi konfigurasi komponen yang ada pada setiap perangkat target, atau konfigurasi default komponen jika tidak ada pada perangkat target. Anda dapat menentukan pembaruan konfigurasi berikut:
  - Pembaruan reset (`reset`) — (Opsional) Daftar pointer JSON yang menentukan nilai konfigurasi untuk di-reset ke nilai default-nya pada perangkat target. Perangkat lunak inti AWS IoT Greengrass memberlakukan pembaruan reset sebelum menerapkan pembaruan merge. Untuk informasi selengkapnya, lihat Pembaruan reset.
  - Pembaruan merge (`merge`) - (Opsional) Sebuah dokumen JSON yang menentukan nilai konfigurasi yang akan digabungkan ke perangkat target. Anda harus membuat serial dokumen JSON sebagai string. Untuk informasi selengkapnya, lihat Pembaruan merge.
-

`runWith`— (Opsional) Opsi proses sistem yang digunakan perangkat lunak AWS IoT Greengrass Core untuk menjalankan proses komponen ini pada perangkat inti. Jika Anda menghilangkan parameter dalam `runWith` objek, perangkat lunak AWS IoT Greengrass Core menggunakan nilai default yang Anda konfigurasikan pada komponen inti Greengrass.

Anda dapat menentukan salah satu opsi berikut:

- `posixUser`— Pengguna sistem POSIX dan, secara opsional, kelompok untuk digunakan untuk menjalankan komponen ini pada perangkat inti Linux. Pengguna, dan grup jika ditentukan, harus ada di setiap perangkat inti Linux. Tentukan pengguna dan grup yang dipisahkan dengan titik dua (`:`) dalam format berikut: `user:group`. Grup ini opsional. Jika Anda tidak menentukan grup, perangkat lunak AWS IoT Greengrass Core menggunakan grup utama untuk pengguna. Untuk informasi selengkapnya, lihat Konfigurasikan pengguna yang menjalankan komponen.
- `windowsUser`— Pengguna Windows yang digunakan untuk menjalankan komponen ini pada perangkat inti Windows. Pengguna harus ada di setiap perangkat inti Windows, dan nama serta kata sandinya harus disimpan dalam instance Credentials Manager LocalSystem akun. Untuk informasi selengkapnya, lihat Konfigurasikan pengguna yang menjalankan komponen.

  Fitur ini tersedia untuk v2.5.0 dan yang lebih baru dari komponen inti Greengrass.
- `systemResourceLimits`— Batas sumber daya sistem untuk diterapkan pada proses komponen ini. Anda dapat menerapkan batas sumber daya sistem ke komponen Lambda generik dan non-kontainer. Untuk informasi selengkapnya, lihat Konfigurasikan batas sumber daya sistem untuk komponen.

  Anda dapat menentukan salah satu opsi berikut:

  - `cpus`— Jumlah maksimum waktu CPU yang dapat digunakan oleh proses komponen ini pada perangkat inti. Total waktu CPU perangkat inti setara dengan jumlah inti CPU perangkat. Misalnya, pada perangkat inti dengan 4 core CPU, Anda dapat mengatur nilai ini 2 untuk membatasi proses komponen ini hingga 50 persen penggunaan setiap inti CPU. Pada perangkat dengan 1 inti CPU, Anda dapat mengatur nilai ini `0.25` untuk membatasi proses komponen ini hingga 25 persen penggunaan CPU. Jika Anda menetapkan nilai ini ke angka yang lebih besar dari jumlah inti CPU, perangkat lunak AWS IoT Greengrass Core tidak membatasi penggunaan CPU komponen.
  - `memory`— Jumlah maksimum RAM (dalam kilobyte) yang dapat digunakan oleh proses komponen ini pada perangkat inti.

Fitur ini tersedia untuk v2.4.0 dan yang lebih baru dari komponen inti Greengrass. AWS IoT Greengrasssaat ini tidak mendukung fitur ini di perangkat inti Windows.

Example Contoh pembaruan konfigurasi dasar

Obejk `components` contoh berikut menentukan untuk men-deploy komponen, `com.example.PythonRuntime`, yang mengharapkan parameter konfigurasi bernama `pythonVersion`.

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName",
  "components": {
    "com.example.PythonRuntime": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"pythonVersion\":\"3.7\"}"
      }
    }
  }
}
```

Example Contoh pembaruan konfigurasi dengan pembaruan reset dan merge

Pertimbangkan komponen contoh dasbor industri, `com.example.IndustrialDashboard`, yang memiliki konfigurasi default berikut.

```
{
  "name": null,
  "mode": "REQUEST",
  "network": {
    "useHttps": true,
    "port": {
      "http": 80,
      "https": 443
    },
  },
  "tags": []
```

```
}
```

Pembaruan konfigurasi berikut menentukan petunjuk berikut:

1. Reset pengaturan HTTPS ke nilai default (`true`).
2. Setel ulang daftar tag industri ke daftar kosong.
3. Gabungkan daftar tag industri yang mengidentifikasi aliran data suhu dan tekanan untuk dua boiler.

```
{
  "reset": [
    "/network/useHttps",
    "/tags"
  ],
  "merge": {
    "tags": [
      "/boiler/1/temperature",
      "/boiler/1/pressure",
      "/boiler/2/temperature",
      "/boiler/2/pressure"
    ]
  }
}
```

Contoh objek `components` berikut menentukan untuk men-deploy komponen dasbor industri ini dan pembaruan konfigurasi.

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "reset": [
          "/network/useHttps",
          "/tags"
        ],
        "merge": "{\"tags\":[\"/boiler/1/temperature\",\"/boiler/1/pressure\",\"/
boiler/2/temperature\",\"/boiler/2/pressure\"]}"
```

```
        }
      }
    }
  }
```

5. (Opsional) Tentukan kebijakan deployment untuk deployment ini. Anda dapat mengonfigurasi ketika perangkat inti dapat dengan aman menerapkan deployment atau apa yang harus dilakukan jika perangkat inti gagal untuk menerapkan deployment tersebut. Untuk melakukannya, tambahkan objek `deploymentPolicies` pada `deployment.json`, lalu lakukan langkah-langkah berikut:

   1. (Opsional) Tentukan kebijakan pembaruan komponen (`componentUpdatePolicy`). Kebijakan ini menentukan apakah deployment memungkinkan komponen menunda pembaruan sampai siap untuk diperbarui. Sebagai contoh, komponen mungkin perlu membersihkan sumber daya atau menyelesaikan tindakan penting sebelum dapat di-restart untuk menerapkan pembaruan. Kebijakan ini juga menentukan jumlah waktu yang harus ditanggapi komponen pada pemberitahuan pembaruan.

      Kebijakan ini merupakan objek dengan parameter berikut:

      - `action` — (Opsional) Apakah akan memberi tahu komponen atau tidak dan menunggunya melaporkan kapan ia siap untuk diperbarui. Pilih dari salah satu pilihan berikut:
        - `NOTIFY_COMPONENTS` – Deployment akan memberitahu setiap komponen sebelum menghentikan dan memperbarui komponen itu. Komponen dapat menggunakan operasi IPC SubscribeToComponentUpdates untuk menerima pemberitahuan ini.
        - `SKIP_NOTIFY_COMPONENTS` – Deployment tidak memberitahu komponen atau menunggunya sampai aman untuk diperbarui.

        Default ke `NOTIFY_COMPONENTS`.
      - `timeoutInSeconds` Jumlah waktu, dalam hitungan detik, ketika setiap komponen merespons notifikasi pembaruan dengan perintah operasi IPC DeferComponentUpdate. Jika komponen tidak merespons dalam jumlah waktu ini, maka deployment akan berlangsung pada perangkat inti.

        Default ke 60 detik.

   2. (Opsional) Tentukan kebijakan validasi konfigurasi (`configurationValidationPolicy`). Kebijakan ini menentukan berapa lama setiap komponen harus memvalidasi pembaruan konfigurasi dari suatu deployment. Komponen dapat menggunakan operasi IPC SubscribeToValidateConfigurationUpdates untuk berlangganan pemberitahuan untuk

pembaruan konfigurasinya sendiri. Kemudian, komponen dapat menggunakan operasi IPC SendConfigurationValidityReport untuk memberitahu perangkat lunak inti AWS IoT Greengrass jika pembaruan konfigurasi valid. Jika pembaruan konfigurasi tidak valid, deployment akan gagal.

Kebijakan ini merupakan objek dengan parameter berikut:

- `timeoutInSeconds` (Opsional) Jumlah waktu, dalam hitungan detik, ketika setiap komponen harus memvalidasi pembaruan konfigurasi. Jika komponen tidak merespons dalam jumlah waktu ini, maka deployment akan berlangsung pada perangkat inti.

  Default ke 30 detik.

3. (Opsional) Tentukan kebijakan penanganan kegagalan (`failureHandlingPolicy`). Kebijakan ini adalah string yang menentukan apakah akan memutar kembali perangkat atau tidak jika deployment tersebut gagal. Pilih dari salah satu pilihan berikut:

   - `ROLLBACK` — Jika deployment gagal pada perangkat inti, maka perangkat lunak inti AWS IoT Greengrass akan memutar kembali perangkat inti ke konfigurasi sebelumnya.
   - `DO_NOTHING` — Jika deployment gagal pada perangkat inti, maka perangkat lunak inti AWS IoT Greengrass akan membuat konfigurasi baru. Hal ini dapat mengakibatkan komponen rusak jika konfigurasi baru tidak valid.

   Default ke `ROLLBACK`.

Deployment Anda di `deployment.json` mungkin terlihat serupa dengan contoh berikut:

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "reset": [
          "/network/useHttps",
          "/tags"
        ],
        "merge": "{\"tags\":[\"/boiler/1/temperature\",\"/boiler/1/pressure\",\"/
boiler/2/temperature\",\"/boiler/2/pressure\"]}"
      }
    }
```

```
  },
  "deploymentPolicies": {
    "componentUpdatePolicy": {
      "action": "NOTIFY_COMPONENTS",
      "timeoutInSeconds": 30
    },
    "configurationValidationPolicy": {
      "timeoutInSeconds": 60
    },
    "failureHandlingPolicy": "ROLLBACK"
  }
}
```

6. (Opsional) Tentukan bagaimana deployment berhenti, keluar, atau habis waktu. AWS IoT Greengrass menggunakan tugas AWS IoT Core untuk mengirim deployment ke perangkat inti, sehingga opsi ini identik dengan opsi konfigurasi untuk tugas AWS IoT Core. Untuk informasi selengkapnya, lihat Peluncuran tugas dan batalkan konfigurasi di Panduan Developer AWS IoT.

   Untuk menentukan pilihan tugas, tambahkan objek `iotJobConfiguration` pada `deployment.json`. Kemudian, tentukan pilihan yang akan dikonfigurasi.

   Deployment Anda di `deployment.json` mungkin terlihat serupa dengan contoh berikut:

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "reset": [
          "/network/useHttps",
          "/tags"
        ],
        "merge": "{\"tags\":[\"/boiler/1/temperature\",\"/boiler/1/pressure\",\"/
boiler/2/temperature\",\"/boiler/2/pressure\"]}"
      }
    }
  },
  "deploymentPolicies": {
    "componentUpdatePolicy": {
      "action": "NOTIFY_COMPONENTS",
      "timeoutInSeconds": 30
```

```
    },
    "configurationValidationPolicy": {
      "timeoutInSeconds": 60
    },
    "failureHandlingPolicy": "ROLLBACK"
  },
  "iotJobConfiguration": {
    "abortConfig": {
      "criteriaList": [
        {
          "action": "CANCEL",
          "failureType": "ALL",
          "minNumberOfExecutedThings": 100,
          "thresholdPercentage": 5
        }
      ]
    },
    "jobExecutionsRolloutConfig": {
      "exponentialRate": {
        "baseRatePerMinute": 5,
        "incrementFactor": 2,
        "rateIncreaseCriteria": {
          "numberOfNotifiedThings": 10,
          "numberOfSucceededThings": 5
        }
      },
      "maximumPerMinute": 50
    },
    "timeoutConfig":  {
      "inProgressTimeoutInMinutes": 5
    }
  }
}
```

7. (Opsional) Tambahkan tag (`tags`) untuk deployment itu. Untuk informasi selengkapnya, lihat Beri tag pada sumber daya AWS IoT Greengrass Version 2 Anda.

8. Jalankan perintah berikut untuk membuat deployment dari `deployment.json`.

```
aws greengrassv2 create-deployment --cli-input-json file://deployment.json
```

Tanggapan meliputi `deploymentId` yang menentukan deployment ini. Anda dapat menggunakan ID deployment untuk memeriksa status deployment. Lihat informasi yang lebih lengkap di Periksa status deployment.

Perbarui konfigurasi komponen

Konfigurasi komponen adalah objek JSON yang menentukan parameter untuk setiap komponen. Resep setiap komponen menentukan konfigurasi default, yang Anda ubah ketika Anda men-deploy komponen ke perangkat inti.

Bila Anda membuat deployment, Anda dapat menentukan pembaruan konfigurasi yang akan diterapkan bagi setiap komponen. Pembaruan konfigurasi adalah operasi patch, yang berarti bahwa pembaruan mengubah konfigurasi komponen yang ada pada perangkat inti. Jika perangkat inti tidak memiliki komponen, maka pembaruan konfigurasi akan memodifikasi dan menerapkan konfigurasi default untuk deployment tersebut.

Pembaruan konfigurasi menentukan pembaruan reset dan pembaruan merge. Pembaruan reset menentukan nilai konfigurasi yang akan di-reset ke default-nya atau dihapus. Pembaruan merge menentukan nilai konfigurasi baru yang akan ditetapkan untuk komponen. Saat Anda menerapkan pembaruan konfigurasi, perangkat lunak AWS IoT Greengrass Core menjalankan pembaruan reset sebelum pembaruan gabungan.

Komponen dapat memvalidasi pembaruan konfigurasi yang Anda deploy. Komponen berlangganan untuk menerima pemberitahuan ketika deployment mengubah konfigurasi, dan dapat menolak konfigurasi yang tidak mendukung. Untuk informasi selengkapnya, lihat Berinteraksilah dengan konfigurasi komponen.

Topik

- Pembaruan reset
- Pembaruan merge
- Contoh

Pembaruan reset

Pembaruan reset menentukan nilai konfigurasi yang akan di-reset ke default-nya pada perangkat inti. Jika nilai konfigurasi tidak memiliki nilai default, maka pembaruan reset akan menghapus nilai dari

konfigurasi komponen. Hal ini dapat membantu Anda memperbaiki komponen yang rusak sebagai hasil dari konfigurasi yang tidak valid.

Gunakan daftar pointer JSON untuk menentukan nilai konfigurasi yang akan di-reset. Pointer JSON dimulai dengan garis miring `/`. Untuk menentukan nilai dalam konfigurasi komponen bersusun, gunakan garis miring ke depan (`/`) untuk memisahkan kunci-kunci pada setiap tingkat dalam konfigurasi. Untuk informasi selengkapnya, lihat spesifikasi pointer JSON.

> **Note**
>
> Anda hanya dapat mengatur ulang seluruh daftar ke nilai default. Anda tidak dapat menggunakan pembaruan reset untuk mengatur ulang elemen individual dalam daftar.

Untuk me-reset seluruh konfigurasi komponen untuk nilai default, tentukan string kosong tunggal sebagai pembaruan reset.

```
"reset": [""]
```

Pembaruan merge

Pembaruan merge menentukan nilai konfigurasi yang akan dimasukkan ke konfigurasi komponen pada inti. Pembaruan gabungan adalah objek JSON yang digabungkan oleh perangkat lunak AWS IoT Greengrass Core setelah menyetel ulang nilai di jalur yang Anda tentukan dalam pembaruan reset. Saat Anda menggunakan AWS CLI atau AWS SDK, Anda harus membuat serial objek JSON ini sebagai string.

Anda dapat menggabungkan pasangan kunci-nilai yang tidak ada dalam konfigurasi default komponen. Anda juga dapat menggabungkan pasangan kunci-nilai yang memiliki jenis yang berbeda dari nilai dengan kunci yang sama. Nilai baru akan menggantikan nilai lama. Ini berarti bahwa Anda dapat mengubah struktur objek konfigurasi.

Anda dapat menggabungkan nilai-nilai null dan string, daftar, dan objek kosong.

> **Note**
>
> Anda tidak dapat menggunakan pembaruan merge untuk tujuan memasukkan atau menambahkan elemen ke daftar. Anda dapat mengganti seluruh daftar, atau Anda dapat menentukan objek di mana setiap elemen memiliki kunci yang unik.

AWS IoT Greengrass menggunakan JSON untuk nilai konfigurasi. JSON menentukan jenis nomor tetapi tidak membedakan antara bilangan bulat dan float. Akibatnya, nilai konfigurasi mungkin berubah menjadi float di AWS IoT Greengrass. Untuk memastikan bahwa komponen Anda menggunakan jenis data yang benar, kami sarankan Anda menentukan nilai konfigurasi numerik sebagai string. Kemudian, buat komponen Anda mengurainya sebagai bilangan bulat atau float. Hal ini akan memastikan bahwa nilai konfigurasi Anda memiliki jenis yang sama dalam konfigurasi dan pada perangkat inti Anda.

Gunakan variabel resep dalam menggabungkan pembaruan

Fitur ini tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass.

Jika Anda menyetel opsi konfigurasi ComponentConfiguration interpolasi inti Greengrass `true` ke, Anda dapat menggunakan variabel resep, selain variabel resep, dalam pembaruan gabungan. *`component_dependency_name`*`:configuration:`*`json_pointer`* Misalnya, Anda dapat menggunakan variabel `{iot:thingName}` resep dalam pembaruan gabungan untuk menyertakan nama AWS IoT benda perangkat inti dalam nilai konfigurasi komponen, seperti kebijakan otorisasi komunikasi antarproses (IPC).

Contoh

Contoh berikut menunjukkan pembaruan konfigurasi untuk komponen dasbor yang memiliki konfigurasi default berikut. Contoh komponen ini menampilkan informasi tentang peralatan industri.

```
{
  "name": null,
  "mode": "REQUEST",
  "network": {
    "useHttps": true,
    "port": {
      "http": 80,
      "https": 443
    },
  },
  "tags": []
}
```

Resep komponen dasbor industri

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.IndustrialDashboard",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Displays information about industrial equipment.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "name": null,
      "mode": "REQUEST",
      "network": {
        "useHttps": true,
        "port": {
          "http": 80,
          "https": 443
        },
      },
      "tags": []
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "run": "python3 -u {artifacts:path}/industrial_dashboard.py"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "run": "py -3 -u {artifacts:path}/industrial_dashboard.py"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.IndustrialDashboard
ComponentVersion: '1.0.0'
ComponentDescription: Displays information about industrial equipment.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    name: null
    mode: REQUEST
    network:
      useHttps: true
      port:
        http: 80
        https: 443
    tags: []
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      run: |
        python3 -u {artifacts:path}/industrial_dashboard.py
  - Platform:
      os: windows
    Lifecycle:
      run: |
        py -3 -u {artifacts:path}/industrial_dashboard.py
```

Example Contoh 1: Gabungkan pembaruan

Anda membuat penerapan yang menerapkan pemutakhiran konfigurasi berikut, yang menentukan pembaruan gabungan tetapi bukan pembaruan reset. Pembaruan konfigurasi ini memberi tahu komponen untuk menampilkan dasbor pada port HTTP 8080 dengan data dari dua boiler.

Console

Konfigurasi untuk digabungkan

```
{
  "name": "Factory 2A",
```

```
  "network": {
    "useHttps": false,
    "port": {
      "http": 8080
    }
  },
  "tags": [
    "/boiler/1/temperature",
    "/boiler/1/pressure",
    "/boiler/2/temperature",
    "/boiler/2/pressure"
  ]
}
```

AWS CLI

Perintah berikut membuat penyebaran ke perangkat inti.

```
aws greengrassv2 create-deployment --cli-input-json file://dashboard-deployment.json
```

`dashboard-deployment.json`File berisi dokumen JSON berikut.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"name\":\"Factory 2A\",\"network\":{\"useHttps\":false,\"port
\":{\"http\":8080}},\"tags\":[\"/boiler/1/temperature\",\"/boiler/1/pressure\",\"/
boiler/2/temperature\",\"/boiler/2/pressure\"]}"
      }
    }
  }
}
```

Greengrass CLI

Perintah Greengrass CLI berikut membuat penerapan lokal pada perangkat inti.

```
sudo greengrass-cli deployment create \
```

```
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.IndustrialDashboard=1.0.0" \
  --update-config dashboard-configuration.json
```

`dashboard-configuration.json`File berisi dokumen JSON berikut.

```
{
  "com.example.IndustrialDashboard": {
    "MERGE": {
      "name": "Factory 2A",
      "network": {
        "useHttps": false,
        "port": {
          "http": 8080
        }
      },
      "tags": [
        "/boiler/1/temperature",
        "/boiler/1/pressure",
        "/boiler/2/temperature",
        "/boiler/2/pressure"
      ]
    }
  }
}
```

Setelah pembaruan ini, komponen dasbor akan memiliki konfigurasi berikut.

```
{
  "name": "Factory 2A",
  "mode": "REQUEST",
  "network": {
    "useHttps": false,
    "port": {
      "http": 8080,
      "https": 443
    }
  },
  "tags": [
    "/boiler/1/temperature",
    "/boiler/1/pressure",
```

```
    "/boiler/2/temperature",
    "/boiler/2/pressure"
  ]
}
```

Example Contoh 2: Setel ulang dan gabungkan pembaruan

Kemudian, Anda membuat penerapan yang menerapkan pembaruan konfigurasi berikut, yang menentukan pembaruan reset dan pembaruan gabungan. Pembaruan ini menentukan untuk menampilkan dasbor pada port HTTPS default dengan data dari boiler yang berbeda. Pembaruan ini mengubah konfigurasi yang dihasilkan dari pembaruan konfigurasi pada contoh sebelumnya.

Console

Setel ulang jalur

```
[
  "/network/useHttps",
  "/tags"
]
```

Konfigurasi untuk digabungkan

```
{
  "tags": [
    "/boiler/3/temperature",
    "/boiler/3/pressure",
    "/boiler/4/temperature",
    "/boiler/4/pressure"
  ]
}
```

AWS CLI

Perintah berikut membuat penyebaran ke perangkat inti.

```
aws greengrassv2 create-deployment --cli-input-json file://dashboard-
deployment2.json
```

`dashboard-deployment2.json`File berisi dokumen JSON berikut.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "reset": [
          "/network/useHttps",
          "/tags"
        ],
        "merge": "{\"tags\":[\"/boiler/3/temperature\",\"/boiler/3/pressure\",\"/
boiler/4/temperature\",\"/boiler/4/pressure\"]}"
      }
    }
  }
}
```

Greengrass CLI

Perintah Greengrass CLI berikut membuat penerapan lokal pada perangkat inti.

```
sudo greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.IndustrialDashboard=1.0.0" \
  --update-config dashboard-configuration2.json
```

`dashboard-configuration2.json`File berisi dokumen JSON berikut.

```
{
  "com.example.IndustrialDashboard": {
    "RESET": [
      "/network/useHttps",
      "/tags"
    ],
    "MERGE": {
      "tags": [
        "/boiler/3/temperature",
        "/boiler/3/pressure",
        "/boiler/4/temperature",
        "/boiler/4/pressure"
```

```
        ]
      }
    }
}
```

Setelah pembaruan ini, komponen dasbor akan memiliki konfigurasi berikut.

```
{
  "name": "Factory 2A",
  "mode": "REQUEST",
  "network": {
    "useHttps": true,
    "port": {
      "http": 8080,
      "https": 443
    }
  },
  "tags": [
    "/boiler/3/temperature",
    "/boiler/3/pressure",
    "/boiler/4/temperature",
    "/boiler/4/pressure",
  ]
}
```

Buat subdeployments

> **Note**
>
> Fitur subdeployment tersedia di Greengrass nucleus versi 2.9.0 dan yang lebih baru. Tidak mungkin untuk menerapkan konfigurasi ke subdeployment dengan versi komponen inti Greengrass sebelumnya.

Subdeployment adalah penerapan yang menargetkan subset perangkat yang lebih kecil dalam penerapan induk. Anda dapat menggunakan subdeployments untuk menerapkan konfigurasi ke subset perangkat yang lebih kecil. Anda juga dapat membuat subdeployment untuk mencoba kembali penerapan induk yang gagal ketika satu atau beberapa perangkat dalam penerapan induk tersebut gagal. Dengan fitur ini, Anda dapat memilih perangkat yang gagal dalam penerapan induk tersebut

dan membuat subdeployment untuk menguji konfigurasi hingga subdeployment berhasil. Setelah subdeployment berhasil, Anda dapat menerapkan ulang konfigurasi tersebut ke penerapan induk.

Ikuti langkah-langkah di bagian ini untuk membuat subdeployment dan memeriksa statusnya. Untuk informasi selengkapnya tentang cara membuat penerapan, lihat Membuat penerapan.

Untuk membuat subdeployment () AWS CLI

1. Jalankan perintah berikut untuk mengambil penerapan terbaru untuk grup sesuatu. Ganti ARN dalam perintah tersebut dengan ARN dari grup objek untuk kueri. Setel `--history-filter` **`LATEST_ONLY`** untuk melihat penerapan terbaru dari grup benda itu.

   ```
   aws greengrassv2 list-deployments --target-arn arn:aws:iot:region:account-
   id:thinggroup/thingGroupName --history-filter LATEST_ONLY
   ```

2. Salin `deploymentId` dari respons ke list-deployments perintah untuk digunakan pada langkah berikutnya.
3. Jalankan perintah berikut untuk mengambil status deployment. Ganti *`deploymentId`* dengan ID penyebaran untuk kueri.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

4. Salin `iotJobId` dari respons ke get-deployment perintah yang akan digunakan pada langkah berikut.
5. Jalankan perintah berikut untuk mengambil daftar eksekusi pekerjaan untuk pekerjaan yang ditentukan. Ganti *`JoBid`* dengan `iotJobId` dari langkah sebelumnya. Ganti *`status`* dengan status yang ingin Anda filter. Anda dapat memfilter hasil dengan status berikut:

   - `QUEUED`
   - `IN_PROGRESS`
   - `SUCCEEDED`
   - `FAILED`
   - `TIMED_OUT`
   - `REJECTED`
   - `REMOVED`
   - `CANCELED`

```
aws iot list-job-executions-for-job --job-id jobID --status status
```

6. Buat grup AWS IoT hal baru, atau gunakan grup hal yang sudah ada, untuk subdeployment Anda. Kemudian, tambahkan AWS IoT sesuatu ke grup benda ini. Anda menggunakan grup objek untuk mengelola armada perangkat inti Greengrass. Saat menerapkan komponen perangkat lunak ke perangkat, Anda dapat menargetkan perangkat individual atau grup perangkat. Anda dapat menambahkan perangkat ke grup benda dengan penerapan Greengrass aktif. Setelah ditambahkan, Anda kemudian dapat menyebarkan komponen perangkat lunak grup benda itu ke perangkat itu.

   Untuk membuat grup hal baru dan menambahkan perangkat Anda ke dalamnya, lakukan hal berikut:

   a. Buat grup AWS IoT hal. Ganti *MyGreengrassCoreGroup*dengan nama untuk grup hal baru. Anda tidak dapat menggunakan titik dua (:) dalam nama grup benda.

      > **Note**
      >
      > Jika grup benda untuk subdeployment digunakan dengan satu`parentTargetArn`, itu tidak dapat digunakan kembali dengan armada induk yang berbeda. Jika grup sesuatu telah digunakan untuk membuat subdeployment untuk armada lain, API akan mengembalikan kesalahan.

      ```
      aws iot create-thing-group --thing-group-name MyGreengrassCoreGroup
      ```

      Jika permintaan berhasil, responsnya terlihat mirip dengan contoh berikut:

      ```
      {
        "thingGroupName": "MyGreengrassCoreGroup",
        "thingGroupArn": "arn:aws:iot:us-
      west-2:123456789012:thinggroup/MyGreengrassCoreGroup",
        "thingGroupId": "4df721e1-ff9f-4f97-92dd-02db4e3f03aa"
      }
      ```

   b. Tambahkan inti Greengrass yang disediakan ke grup hal Anda. Jalankan perintah berikut dengan parameter ini:

- Ganti *MyGreengrassCore*dengan nama inti Greengrass yang Anda sediakan.
- Ganti *MyGreengrassCoreGroup*dengan nama grup barang Anda.

```
aws iot add-thing-to-thing-group --thing-name MyGreengrassCore --thing-group-
name MyGreengrassCoreGroup
```

Perintah tersebut tidak memiliki output apa pun jika permintaan berhasil.

7. Buat file bernama `deployment.json`, lalu salin objek JSON berikut ke dalam file. Ganti *targetArn* dengan ARN dari grup hal yang akan ditargetkan AWS IoT untuk subdeployment. Target subdeployment hanya bisa berupa grup benda. ARN grup benda memiliki format berikut:

   - Kelompok hal — `arn:aws:iot:`*region*`:`*account-id*`:thinggroup/`*thingGroupName*

```
{
  "targetArn": "targetArn"
}
```

8. Jalankan perintah berikut lagi untuk mendapatkan detail penerapan asli. Detail ini mencakup metadata, komponen, dan konfigurasi pekerjaan. Ganti *DeploymentID* dengan ID dari. Step 1 Anda dapat menggunakan konfigurasi penerapan ini untuk mengonfigurasi subdeployment dan membuat perubahan sesuai kebutuhan.

```
aws greengrassv2 get-deployment --deployment-id deploymentId
```

Tanggapan berisi detail deployment. Salin salah satu pasangan kunci-nilai berikut dari respons get-deployment perintah ke dalam. `deployment.json` Anda dapat mengubah nilai-nilai ini untuk subdeployment. Untuk informasi selengkapnya tentang detail perintah ini, lihat GetDeployment.

- `components`— Komponen penyebaran. Untuk menghapus komponen, hapus dari objek ini.
- `deploymentName`— Nama penyebaran.
- `deploymentPolicies`— Kebijakan penyebaran.
- `iotJobConfiguration`— Konfigurasi pekerjaan penerapan.
- `parentTargetArn`— Target penyebaran induk.

- `tags`— Tag penyebaran.

9. Jalankan perintah berikut untuk membuat subdeployment dari. `deployment.json` Ganti *`subDeploymentName dengan nama`* untuk subdeployment.

```
aws greengrassv2 create-deployment --deployment-name subdeploymentName --cli-input-
json file://deployment.json
```

Responsnya mencakup a `deploymentId` yang mengidentifikasi subdeployment ini. Anda dapat menggunakan ID deployment untuk memeriksa status deployment. Untuk informasi selengkapnya, lihat Memeriksa status penerapan.

10. Jika subdeployment berhasil, Anda dapat menggunakan konfigurasinya untuk merevisi penerapan induk. Salin `deployment.json` yang Anda gunakan pada langkah sebelumnya. Ganti `targetArn` dalam file JSON dengan ARN penerapan induk dan jalankan perintah berikut untuk membuat penyebaran induk menggunakan konfigurasi baru ini.

> **Note**
>
> Jika Anda membuat revisi penerapan baru dari armada induk, ini akan menggantikan semua revisi penerapan dan subdeployment untuk penerapan induk tersebut. Untuk informasi selengkapnya, lihat Merevisi penerapan.

```
aws greengrassv2 create-deployment --cli-input-json file://deployment.json
```

Tanggapan meliputi `deploymentId` yang menentukan deployment ini. Anda dapat menggunakan ID deployment untuk memeriksa status deployment. Lihat informasi yang lebih lengkap di Periksa status deployment.

Revisi deployment

Setiap objek target atau grup objek dapat memiliki satu deployment pada satu waktu. Ketika Anda membuat deployment untuk target yang sudah memiliki deployment, komponen perangkat lunak dalam deployment baru akan menggantikan komponen dari deployment sebelumnya. Jika deployment baru tidak menentukan komponen yang ditetapkan oleh deployment sebelumnya, perangkat lunak inti AWS IoT Greengrass akan menghapus komponen dari perangkat inti target.

Anda dapat merevisi deployment yang ada sehingga Anda tidak menghapus komponen yang berjalan di perangkat inti dari deployment sebelumnya ke target.

Untuk merevisi penerapan, Anda membuat penerapan yang dimulai dari komponen dan konfigurasi yang sama yang ada di penerapan sebelumnya. Anda menggunakan CreateDeploymentoperasi, yang merupakan operasi yang sama yang Anda gunakan untuk membuat penerapan.

Untuk merevisi deployment (AWS CLI)

1. Jalankan perintah berikut untuk membuat daftar penerapan untuk target penyebaran. Ganti *targetArn* dengan ARN target objek AWS IoT atau grup objek.

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   Tanggapan berisi daftar dengan deployment terbaru untuk target. Salin `deploymentId` dari respons untuk digunakan di langkah berikutnya.

   > **Note**
   >
   > Anda juga dapat merevisi deployment selain revisi terbaru untuk target. Tentukan `--history-filter ALL` untuk mencantumkan semua deployment untuk target. Kemudian, salin ID penerapan yang ingin Anda revisi.

2. Jalankan perintah berikut untuk mendapatkan detail penerapan. Detail ini mencakup metadata, komponen, dan konfigurasi pekerjaan. Ganti *deploymentId* dengan ID dari langkah sebelumnya.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   Tanggapan berisi detail deployment.

3. Buat file bernama `deployment.json` dan salin respons perintah sebelumnya ke dalam file.
4. Hapus pasangan nilai kunci berikut dari objek JSON di `deployment.json`:

   - `deploymentId`
   - `revisionId`
   - `iotJobId`
   - `iotJobArn`

- `creationTimestamp`
- `isLatestForTarget`
- `deploymentStatus`

CreateDeploymentOperasi mengharapkan muatan dengan struktur berikut.

```
{
  "targetArn": "String",
  "components": Map of components,
  "deploymentPolicies": DeploymentPolicies,
  "iotJobConfiguration": DeploymentIoTJobConfiguration,
  "tags": Map of tags
}
```

5. Pada `deployment.json`, lakukan langkah-langkah berikut:
   - Ubah nama deployment ini (`deploymentName`).
   - Ubah komponen deployment ini (`components`).
   - Ubah kebijakan deployment ini (`deploymentPolicies`).
   - Ubah konfigurasi tugas deployment (`iotJobConfiguration`).
   - Ubah tag deployment ini (`tags`).

   Untuk informasi lebih lanjut tentang cara menentukan detail deployment ini, lihat Buat deployment.
6. Jalankan perintah berikut untuk membuat deployment dari `deployment.json`.

```
aws greengrassv2 create-deployment --cli-input-json file://deployment.json
```

   Tanggapan meliputi `deploymentId` yang menentukan deployment ini. Anda dapat menggunakan ID deployment untuk memeriksa status deployment. Lihat informasi yang lebih lengkap di Periksa status deployment.

Batalkan deployment

Anda dapat membatalkan deployment untuk mencegah komponen perangkat lunaknya diinstal pada perangkat inti AWS IoT Greengrass. Jika Anda membatalkan deployment yang menargetkan grup objek, perangkat inti yang ditambahkan ke grup tidak akan menerima deployment berkelanjutan

tersebut. Jika perangkat inti telah menjalankan deployment, Anda tidak akan mengubah komponen di perangkat tersebut saat membatalkan deployment. Anda harus membuat deployment baru atau merevisi deployment ini untuk mengubah komponen yang berjalan pada perangkat inti yang menerima deployment yang dibatalkan.

Untuk membatalkan deployment (AWS CLI)

1. Jalankan perintah berikut untuk menemukan ID revisi deployment terbaru untuk target. Revisi terbaru adalah satu-satunya deployment yang dapat aktif untuk target, karena deployment sebelumnya batal ketika Anda membuat revisi baru. Ganti *targetArn* dengan ARN target objek AWS IoT atau grup objek.

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   Tanggapan berisi daftar dengan deployment terbaru untuk target. Salin `deploymentId` dari respons untuk digunakan di langkah berikutnya.
2. Jalankan perintah berikut untuk membatalkan deployment. Ganti *deploymentId* dengan ID dari langkah sebelumnya.

   ```
   aws greengrassv2 cancel-deployment --deployment-id deploymentId
   ```

   Jika operasi berhasil, status deployment akan berubah menjadi `CANCELED`.

Periksa status deployment

Anda dapat memeriksa status deployment yang Anda buat di AWS IoT Greengrass. Anda juga dapat memeriksa status tugas AWS IoT yang meluncurkan deployment ke setiap perangkat inti. Sementara deployment aktif, status tugas AWS IoT adalah `IN_PROGRESS`. Setelah Anda membuat revisi baru dari deployment, status tugas AWS IoT revisi sebelumnya berubah ke `CANCELLED`.

Topik

- Periksa status deployment
- Periksa status deployment perangkat

Periksa status deployment

Anda dapat memeriksa status deployment yang Anda identifikasi berdasarkan target atau ID-nya.

Untuk memeriksa status deployment berdasarkan target (AWS CLI)

- Jalankan perintah berikut untuk mengambil status deployment terbaru untuk target. Ganti *targetArn* dengan Amazon Resource Name (ARN) dari grupAWS IoT objek atau grup objek yang ditargetkan deployment.

  ```
  aws greengrassv2 list-deployments --target-arn targetArn
  ```

  Tanggapan berisi daftar dengan deployment terbaru untuk target. Objek deployment ini mencakup status deployment.

Untuk memeriksa status deployment berdasarkan ID (AWS CLI)

- Jalankan perintah berikut untuk mengambil status deployment. Ganti *deploymentId* dengan ID deployment untuk kueri.

  ```
  aws greengrassv2 get-deployment --deployment-id deploymentId
  ```

  Tanggapan berisi status deployment.

Periksa status deployment perangkat

Anda dapat memeriksa status tugas deployment yang berlaku pada perangkat inti individu. Anda juga dapat memeriksa status deployment untuk deployment objek deployment.

Untuk memeriksa status pekerjaan penerapan untuk perangkat inti (AWS CLI)

- Jalankan perintah berikut untuk mengambil status semua deployment untuk perangkat inti. Ganti *coreDeviceName*dengan nama perangkat core yang akan kueri.

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

  Tanggapan berisi daftar tugas deployment untuk perangkat inti. Anda dapat mengidentifikasi tugas untuk deployment berdasarkan tugas`deploymentId` atau`targetArn`. Setiap tugas deployment berisi status tugas pada perangkat inti.

Untuk memeriksa status penyebaran untuk grup hal (AWS CLI)

1. Jalankan perintah berikut untuk mengambil ID deployment yang ada. Ganti *targetArn* dengan ARN objek objek objek.

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   Tanggapan berisi daftar dengan deployment terbaru untuk target. Salin `deploymentId` dari respons untuk digunakan di langkah berikutnya.

   > **Note**
   >
   > Anda juga dapat mencantumkan deployment selain deployment terbaru untuk target. Tentukan `--history-filter ALL` untuk mencantumkan semua deployment untuk target. Kemudian, salin ID deployment yang ingin Anda periksa statusnya.

2. Jalankan perintah berikut ini untuk mendapatkan detail deployment. Ganti *deployment dengan ID* dari langkah sebelumnya.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   Tanggapan berisi informasi tentang deployment. Salin`iotJobId` dari respons yang akan digunakan pada langkah berikut.

3. Jalankan perintah berikut untuk menjelaskan eksekusi tugas core untuk deployment. Ganti *iotJobId*dan *coreDeviceThingNama* dengan ID pekerjaan dari langkah sebelumnya dan perangkat inti yang ingin Anda periksa statusnya.

   ```
   aws iot describe-job-execution --job-id iotJobId --thing-name coreDeviceThingName
   ```

   Tanggapan berisi status deployment dan detail tentang status deployment. Itu`detailsMap` berisi informasi berikut:

   - `detailed-deployment-status`— Status deployment, yang bisa jadi salah satu dari yang berikut:
     - `SUCCESSFUL`- Penyebaran berhasil.
     - `FAILED_NO_STATE_CHANGE`- Penyebaran gagal saat perangkat inti siap untuk menerapkan penyebaran.

- `FAILED_ROLLBACK_NOT_REQUESTED`- Penyebaran gagal, dan penyebaran tidak menentukan untuk memutar kembali ke konfigurasi kerja sebelumnya, sehingga perangkat inti mungkin tidak berfungsi dengan benar.
- `FAILED_ROLLBACK_COMPLETE`- Penyebaran gagal, dan perangkat inti berhasil digulung kembali ke konfigurasi kerja sebelumnya.
- `FAILED_UNABLE_TO_ROLLBACK`- Penyebaran gagal, dan perangkat inti gagal memutar kembali ke konfigurasi kerja sebelumnya, sehingga perangkat inti mungkin tidak berfungsi dengan benar.

  Jika penyebaran gagal, periksa`deployment-failure-cause` nilai dan file log perangkat inti untuk mengidentifikasi masalah. Untuk informasi lebih lanjut tentang cara mengakses file log perangkat core, lihatMemantau AWS IoT Greengrass log.

- `deployment-failure-cause`- Pesan kesalahan yang memberikan rincian tambahan tentang mengapa eksekusi pekerjaan gagal.

Tanggapan tersebut serupa dengan contoh berikut ini.


```
{
  "execution": {
    "jobId": "2cc2698a-5175-48bb-adf2-1dd345606ebd",
    "status": "FAILED",
    "statusDetails": {
      "detailsMap": {
        "deployment-failure-cause": "No local or cloud component version
 satisfies the requirements. Check whether the version constraints conflict
 and that the component exists in your Akun AWS with a version that matches the
 version constraints. If the version constraints conflict, revise deployments
 to resolve the conflict. Component com.example.HelloWorld version constraints:
 LOCAL_DEPLOYMENT requires =1.0.0, thinggroup/MyGreengrassCoreGroup requires
 =1.0.1.",
        "detailed-deployment-status": "FAILED_NO_STATE_CHANGE"
      }
    },
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
    "queuedAt": "2022-02-15T14:45:53.098000-08:00",
    "startedAt": "2022-02-15T14:46:05.670000-08:00",
    "lastUpdatedAt": "2022-02-15T14:46:20.892000-08:00",
    "executionNumber": 1,
    "versionNumber": 3
```

```
  }
}
```

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 manakah yang akan Anda pantau?
- Seberapa seringkah Anda akan memantau sumber daya ini?
- Apa sajakah alat pemantauan yang akan Anda gunakan?
- Siapakah yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu saat terjadi kesalahan?

Topik

Alat-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 mengotomatiskan tugas-tugas pemantauan sebanyak mungkin.

Anda dapat menggunakan alat pemantauan otomatis berikut untuk memantau AWS IoT Greengrass dan melaporkan masalah:

- Amazon CloudWatch Logs — Pantau, simpan, dan akses file log Anda dari AWS CloudTrail atau sumber lain. Untuk informasi selengkapnya, lihat Memantau file log di Panduan CloudWatch Pengguna Amazon.
- AWS CloudTrailPemantauan Log - Bagikan file log antar akun, pantau file CloudTrail log secara real time dengan mengirimkannya ke CloudWatch Log, menulis aplikasi pemrosesan log di Java, dan validasi bahwa file log Anda tidak berubah setelah pengiriman oleh CloudTrail. Untuk informasi selengkapnya, lihat Bekerja dengan file CloudTrail log di Panduan AWS CloudTrail Pengguna.
- Telemetri kesehatan sistem Greengrass — Berlangganlah untuk menerima data telemetri yang dikirim dari inti Greengrass. Untuk informasi selengkapnya, lihat the section called “Kumpulkan data telemetri kondisi sistem”.
- Pemberitahuan kesehatan perangkat Buat acara menggunakan Amazon EventBridge untuk menerima pembaruan status terkait penerapan dan komponen. Untuk informasi selengkapnya, lihat Dapatkan pemberitahuan status kesehatan penerapan dan komponen.
- Layanan status armada — Gunakan operasi API status armada untuk memeriksa status perangkat inti dan komponen Greengrass mereka. Anda juga dapat melihat informasi status armada di AWS IoT Greengrass konsol. Lihat informasi yang lebih lengkap di Periksa status perangkat inti Greengrass.

Memantau 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 Logs dan ke sistem file lokal perangkat inti. Komponen Greengrass yang berjalan pada perangkat inti juga dapat menulis log CloudWatch 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.

Secara default, perangkat lunak AWS IoT Greengrass Core menulis log hanya ke sistem file lokal. Anda dapat melihat log sistem file secara real time, sehingga Anda dapat men-debug komponen Greengrass yang Anda kembangkan dan terapkan. Anda juga dapat mengonfigurasi perangkat inti untuk menulis CloudWatch log ke Log, sehingga Anda dapat memecahkan masalah perangkat inti tanpa akses ke sistem file lokal. Untuk informasi selengkapnya, lihat Aktifkan pencatatan ke CloudWatch Log.

Topik

- Akses log sistem file

- Akses CloudWatch Log
- Akses log layanan sistem
- Aktifkan pencatatan ke CloudWatch Log
- Konfigurasi pencatatan untuk AWS IoT Greengrass
- Log AWS CloudTrail

Akses log sistem file

Perangkat lunak AWS IoT Greengrass Core menyimpan log di */greengrass/v2*/logs folder pada perangkat inti, di */greengrass/v2* mana jalur ke folder AWS IoT Greengrass root. Folder log memiliki struktur berikut.

```
/greengrass/v2
### logs
    ### greengrass.log
    ### greengrass_2021_09_14_15_0.log
    ### ComponentName.log
    ### ComponentName_2021_09_14_15_0.log
    ### main.log
```

- `greengrass.log`— File log perangkat lunak AWS IoT Greengrass Core. Gunakan file log ini untuk melihat informasi real-time tentang komponen dan penerapan. File log ini mencakup log untuk inti Greengrass, yang merupakan inti dari AWS IoT Greengrass perangkat lunak Core, dan komponen plugin, seperti pengelola log dan manajer rahasia.
- *ComponentName*`.log`— File log komponen Greengrass. Gunakan file log komponen untuk melihat informasi real-time tentang komponen Greengrass yang berjalan pada perangkat inti. Komponen generik dan komponen Lambda menulis output standar (stdout) dan kesalahan standar (stderr) ke file log ini.
- `main.log`— File log untuk `main` layanan yang menangani siklus hidup komponen. File log ini akan selalu kosong.

Untuk informasi selengkapnya tentang perbedaan antara komponen plugin, generik, dan Lambda, lihat. Jenis komponen

Pertimbangan berikut berlaku saat Anda menggunakan log sistem file:

- Izin pengguna root

Anda harus memiliki izin root untuk membaca AWS IoT Greengrass log pada sistem file.

- Rotasi file log

  Perangkat lunak AWS IoT Greengrass Core memutar file log setiap jam atau ketika mereka melebihi batas ukuran file. File log yang diputar berisi stempel waktu dalam nama file mereka. Misalnya, file log perangkat lunak AWS IoT Greengrass Core yang diputar mungkin diberi nama`greengrass_2021_09_14_15_0.log`. Batas ukuran file default adalah 1.024 KB (1 MB). Anda dapat mengonfigurasi batas ukuran file pada komponen inti Greengrass.

- Penghapusan file log

  Perangkat lunak AWS IoT Greengrass Core membersihkan file log sebelumnya ketika ukuran file log perangkat lunak AWS IoT Greengrass Core atau file log komponen Greengrass, termasuk file log yang diputar, melebihi batas ruang disk. Batas ruang disk default untuk log perangkat lunak AWS IoT Greengrass Core dan setiap log komponen adalah 10.240 KB (10 MB). Anda dapat mengonfigurasi batas ruang disk log perangkat lunak AWS IoT Greengrass Core pada komponen inti Greengrass atau komponen pengelola log. Anda dapat mengonfigurasi batas ruang disk log setiap komponen pada komponen pengelola log.

Untuk melihat file log perangkat lunak AWS IoT Greengrass inti

- Jalankan perintah berikut untuk melihat file log secara real time. Ganti */greengrass/v2* dengan jalur ke folder AWS IoT Greengrass root.

  Linux or Unix

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

  Windows Command Prompt (CMD)

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

  `type`Perintah menulis konten file ke terminal. Jalankan perintah ini beberapa kali untuk mengamati perubahan dalam file.

  PowerShell

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

Untuk melihat berkas log untuk komponen

- Jalankan perintah berikut untuk melihat file log secara real time. Ganti */greengrass/v2* atau *C:\greengrass\v2* dengan path ke folder AWS IoT Greengrass root, dan ganti *com.example. HelloWorld*dengan nama komponennya.

  Linux or Unix

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

  PowerShell

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

Anda juga dapat menggunakan `logs` perintah CLI Greengrass untuk menganalisis log Greengrass pada perangkat inti. Untuk menggunakan `logs` perintah, Anda harus mengkonfigurasi inti Greengrass untuk mengeluarkan file log format JSON. Lihat informasi yang lebih lengkap di Antarmuka Baris Perintah Greengrass dan log.

Akses CloudWatch Log

Anda dapat menerapkan komponen pengelola log untuk mengonfigurasi perangkat inti untuk menulis ke CloudWatch Log. Untuk informasi selengkapnya, lihat Aktifkan pencatatan ke CloudWatch Log. Kemudian, Anda dapat melihat log di halaman Log di CloudWatch konsol Amazon atau menggunakan API CloudWatch Log.

Nama grup log

```
/aws/greengrass/componentType/region/componentName
```

Nama grup log menggunakan variabel berikut:

- `componentType` — Jenis komponen, yang dapat berupa salah satu dari berikut ini:
  - `GreengrassSystemComponent`— Grup log ini mencakup log untuk komponen inti dan plugin, yang berjalan di JVM yang sama dengan inti Greengrass. Komponen ini merupakan bagian dari inti Greengrass.
  - `UserComponent`— Grup log ini mencakup log untuk komponen generik, komponen Lambda, dan aplikasi lain di perangkat. Komponen ini bukan bagian dari inti Greengrass.

Untuk informasi selengkapnya, lihat Jenis komponen.

- `region` — Wilayah AWS yang menggunakan perangkat inti.
- `componentName` — Nama komponen. Untuk log sistem, nilai ini adalah `System`.

Nama aliran log

```
/date/thing/thingName
```

Nama aliran log menggunakan variabel berikut:

- `date` — Tanggal log, seperti `2020/12/15`. Komponen manajer log menggunakan format `yyyy/MM/dd`.
- `thingName` - Nama perangkat inti.

> **Note**
>
> Jika nama objek berisi titik dua (`:`), manajer log akan menggantikan titik dua dengan tanda tambah (+).

Pertimbangan berikut berlaku saat Anda menggunakan komponen pengelola log untuk menulis ke CloudWatch Log:

- Penundaan log

> **Note**
>
> Kami menyarankan Anda meningkatkan ke pengelola log versi 2.3.0 yang mengurangi penundaan log untuk file log yang diputar dan aktif. Saat Anda meningkatkan ke pengelola log 2.3.0, kami sarankan Anda juga meningkatkan ke Greengrass nucleus 2.9.1.

Komponen pengelola log versi 2.2.8 (dan sebelumnya) memproses dan mengunggah log hanya dari file log yang diputar. Secara default, perangkat lunak AWS IoT Greengrass Core memutar file log setiap jam atau setelah 1.024 KB. Akibatnya, komponen pengelola log mengunggah log hanya setelah perangkat lunak AWS IoT Greengrass Core atau komponen Greengrass menulis log senilai lebih dari 1.024 KB. Anda dapat mengonfigurasi batas ukuran file log yang lebih rendah

untuk menyebabkan file log diputar lebih sering. Hal ini menyebabkan komponen pengelola log mengunggah CloudWatch log ke Log lebih sering.

Komponen pengelola log versi 2.3.0 (dan yang lebih baru) memproses dan mengunggah semua log. Saat Anda menulis log baru, pengelola log versi 2.3.0 (dan yang lebih baru) memproses dan langsung mengunggah file log aktif itu alih-alih menunggu untuk diputar. Ini berarti Anda dapat melihat log baru dalam 5 menit atau kurang.

Komponen pengelola log mengunggah log baru secara berkala. Secara default, komponen pengelola log mengunggah log baru setiap 5 menit. Anda dapat mengonfigurasi interval unggahan yang lebih rendah, sehingga komponen pengelola log mengunggah CloudWatch log ke Log lebih sering dengan mengonfigurasi log. `periodicUploadIntervalSec` Untuk informasi selengkapnya tentang cara mengonfigurasi interval periodik ini, lihat Konfigurasi.

Log dapat diunggah dalam waktu dekat dari sistem file Greengrass yang sama. Jika Anda perlu mengamati log secara real time, pertimbangkan untuk menggunakan log sistem file.

> **Note**
>
> Jika Anda menggunakan sistem file yang berbeda untuk menulis log, pengelola log kembali ke perilaku di komponen pengelola log versi 2.2.8 dan yang lebih lama. Untuk informasi tentang mengakses log sistem file, lihat Mengakses log sistem file.

- Kemiringan jam

  Komponen pengelola log menggunakan proses penandatanganan Signature Version 4 standar untuk membuat permintaan API ke CloudWatch Log. Jika waktu sistem pada perangkat inti tidak sinkron lebih dari 15 menit, maka CloudWatch Log menolak permintaan. Untuk informasi selengkapnya, lihat Proses penandatanganan Versi Tanda Tangan 4 di bagian Referensi Umum AWS.

Akses log layanan sistem

Jika Anda mengonfigurasi perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem, Anda dapat melihat log layanan sistem untuk memecahkan masalah, seperti perangkat lunak yang gagal memulai.

Untuk melihat log layanan sistem (CLI)

1. Jalankan perintah berikut untuk melihat log layanan sistem perangkat lunak AWS IoT Greengrass inti.

   Linux or Unix (systemd)

   ```
   sudo journalctl -u greengrass.service
   ```

   Windows Command Prompt (CMD)

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

   PowerShell

   ```
   gc C:\greengrass\v2\logs\greengrass.wrapper.log
   ```

2. Pada perangkat Windows, perangkat lunak AWS IoT Greengrass Core membuat file log terpisah untuk kesalahan layanan sistem. Jalankan perintah berikut untuk melihat log kesalahan layanan sistem.

   Windows Command Prompt (CMD)

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

   PowerShell

   ```
   gc C:\greengrass\v2\logs\greengrass.err.log
   ```

Pada perangkat Windows, Anda juga dapat menggunakan aplikasi Event Viewer untuk melihat log layanan sistem.

Untuk melihat log layanan Windows (Event Viewer)

1. Buka aplikasi Event Viewer.
2. Pilih Windows Log untuk memperluasnya.
3. Pilih Aplikasi untuk melihat log layanan aplikasi.
4. Temukan dan buka log peristiwa yang Sumbernya greengrass.

Aktifkan pencatatan ke CloudWatch Log

Anda dapat menerapkan komponen pengelola log untuk mengonfigurasi perangkat inti untuk menulis log ke CloudWatch Log. Anda dapat mengaktifkan CloudWatch Log untuk log perangkat lunak AWS IoT Greengrass Inti, dan Anda dapat mengaktifkan CloudWatch Log untuk komponen Greengrass tertentu.

> **Note**
>
> Peran pertukaran token perangkat inti Greengrass harus memungkinkan perangkat inti untuk menulis CloudWatch ke Log, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut. Jika Anda menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya otomatis, perangkat inti Anda memiliki izin ini.
>
> ```
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Action": [
>         "logs:CreateLogGroup",
>         "logs:CreateLogStream",
>         "logs:PutLogEvents",
>         "logs:DescribeLogStreams"
>       ],
>       "Effect": "Allow",
>       "Resource": "arn:aws:logs:*:*:*"
>     }
>   ]
> }
> ```

Untuk mengonfigurasi perangkat inti untuk menulis log perangkat lunak AWS IoT Greengrass Inti ke CloudWatch Log, buat penerapan yang menentukan pembaruan konfigurasi yang disetel `uploadToCloudWatch` ke `true` komponen. `aws.greengrass.LogManager` AWS IoT Greengrass Log perangkat lunak inti menyertakan log untuk inti Greengrass dan komponen plugin.

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true"
```

```
    }
  }
}
```

Untuk mengonfigurasi perangkat inti untuk menulis log komponen Greengrass CloudWatch ke Log, buat penerapan yang menentukan pembaruan konfigurasi yang menambahkan komponen ke daftar konfigurasi logging komponen. Saat Anda menambahkan komponen ke daftar ini, komponen pengelola log akan menulis lognya ke CloudWatch Log. Log komponen termasuk log untuk komponen generik dan komponen Lambda.

```
{
  "logsUploaderConfiguration": {
    "componentLogsConfigurationMap": {
      "com.example.HelloWorld": {

      }
    }
  }
}
```

Saat Anda menerapkan komponen pengelola log, Anda juga dapat mengonfigurasi batas ruang disk dan apakah perangkat inti menghapus file log setelah menulisnya ke CloudWatch Log. Untuk informasi selengkapnya, lihat Konfigurasi pencatatan untuk AWS IoT Greengrass.

Konfigurasi pencatatan untuk AWS IoT Greengrass

Anda dapat mengonfigurasi opsi berikut untuk menyesuaikan logging untuk perangkat inti Greengrass. Untuk mengonfigurasi opsi ini, buat penerapan yang menentukan pembaruan konfigurasi ke inti Greengrass atau komponen pengelola log.

- Menulis log ke CloudWatch Log

  Untuk memecahkan masalah perangkat inti dari jarak jauh, Anda dapat mengonfigurasi perangkat inti untuk menulis perangkat lunak AWS IoT Greengrass inti dan log komponen ke Log. CloudWatch Untuk melakukannya, gunakan dan konfigurasikan komponen pengelola log. Untuk informasi selengkapnya, lihat Aktifkan pencatatan ke CloudWatch Log.
- Menghapus file log yang diunggah

  Untuk mengurangi penggunaan ruang disk, Anda dapat mengonfigurasi perangkat inti untuk menghapus file log setelah menulis file CloudWatch log ke Log. Untuk informasi selengkapnya,

lihat `deleteLogFileAfterCloudUpload` parameter komponen pengelola log, yang dapat Anda tentukan untuk log perangkat lunak AWS IoT Greengrass inti dan log komponen.

- Batas ruang disk log

  Untuk membatasi penggunaan ruang disk, Anda dapat mengonfigurasi ruang disk maksimum untuk setiap log, termasuk file log yang diputar, pada perangkat inti. Misalnya, Anda dapat mengonfigurasi ruang disk gabungan maksimum untuk `greengrass.log` dan `greengrass.log` file yang diputar. Untuk informasi selengkapnya, lihat parameter komponen inti Greengrass dan parameter komponen `logging.totalLogsSizeKBdiskSpaceLimit` pengelola log, yang dapat Anda tentukan AWS IoT Greengrassuntuk log perangkat lunak inti dan log komponen.

- Batas ukuran file log

  Anda dapat mengonfigurasi ukuran file maksimum untuk setiap file log. Setelah file log melebihi batas ukuran file ini, perangkat lunak AWS IoT Greengrass Core membuat file log baru. Komponen pengelola log versi 2.28 (dan sebelumnya) hanya menulis file log yang diputar ke CloudWatch Log, sehingga Anda dapat menentukan batas ukuran file yang lebih rendah untuk menulis CloudWatch log ke Log lebih sering. Komponen pengelola log versi 2.3.0 (dan yang lebih baru) memproses dan mengunggah semua log alih-alih menunggu mereka diputar. Untuk informasi lebih lanjut, lihat parameter batas ukuran file log komponen inti Greengrass (). `logging.fileSizeKB`

- Tingkat log minimum

  Anda dapat mengonfigurasi level log minimum yang ditulis oleh komponen inti Greengrass ke log sistem file. Misalnya, Anda dapat menentukan log `DEBUG` level untuk membantu pemecahan masalah, atau Anda dapat menentukan log `ERROR` level untuk mengurangi jumlah log yang dibuat oleh perangkat inti. Untuk informasi lebih lanjut, lihat parameter tingkat log komponen inti Greengrass (). `logging.level`

  Anda juga dapat mengonfigurasi tingkat log minimum yang ditulis oleh komponen pengelola CloudWatch log ke Log. Misalnya, Anda dapat menentukan tingkat log yang lebih tinggi untuk mengurangi biaya pencatatan. Untuk informasi selengkapnya, lihat `minimumLogLevel` parameter komponen pengelola log, yang dapat Anda tentukan untuk log perangkat lunak AWS IoT Greengrass inti dan log komponen.

- Interval untuk memeriksa log untuk menulis ke CloudWatch Log

  Untuk menambah atau mengurangi seberapa sering komponen pengelola log menulis CloudWatch log ke Log, Anda dapat mengonfigurasi interval di mana ia memeriksa file log baru untuk ditulis. Misalnya, Anda dapat menentukan interval yang lebih rendah untuk melihat CloudWatch log di Log

lebih cepat daripada yang Anda lakukan dengan interval 5 menit default. Anda dapat menentukan interval yang lebih tinggi untuk mengurangi biaya, karena komponen pengelola log mengumpulkan file log ke permintaan yang lebih sedikit. Untuk informasi selengkapnya, lihat parameter interval unggah komponen pengelola log (`periodicUploadIntervalSec`).

- Format log

  Anda dapat memilih apakah perangkat lunak AWS IoT Greengrass Core menulis log dalam format teks atau JSON. Pilih format teks jika Anda membaca log, atau pilih format JSON jika Anda menggunakan aplikasi untuk membaca atau mengurai log. Untuk informasi lebih lanjut, lihat parameter format log komponen inti Greengrass (). `logging.format`

- Folder log sistem file lokal

  Anda dapat mengubah folder log dari */greengrass/v2*`/logs` folder lain pada perangkat inti. Untuk informasi lebih lanjut, lihat parameter direktori keluaran komponen inti Greengrass (). `logging.outputDirectory`

Log AWS CloudTrail

AWS IoT Greengrassterintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau Layanan AWS dalamAWS IoT Greengrass. Lihat informasi yang lebih lengkap di Log panggilan AWS IoT Greengrass V2 API dengan AWS CloudTrail.

Log panggilan AWS IoT Greengrass V2 API dengan AWS CloudTrail

AWS IoT Greengrass V2 terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di AWS IoT Greengrass Version 2. CloudTrail menangkap semua panggilan API untuk AWS IoT Greengrass sebagai peristiwa. Panggilan yang ditangkap termasuk panggilan dari AWS IoT Greengrass konsol dan panggilan kode ke operasi AWS IoT Greengrass API.

Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket 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 dibuat AWS IoT Greengrass, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk informasi selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

AWS IoT Greengrass V2 informasi di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas terjadi di AWS 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 acara yang sedang berlangsung di Anda Akun AWS, termasuk acara untuk AWS IoT Greengrass, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket S3. Secara default, saat Anda membuat jejak di konsol, jejak berlaku untuk semua Wilayah AWS s. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket 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 V2 tindakan dicatat oleh CloudTrail dan didokumentasikan dalam Referensi AWS IoT Greengrass V2 API. Misalnya, panggilan ke`CreateComponentVersion`, `CreateDeployment` dan `CancelDeployment` 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:

- Apakah permintaan itu dibuat dengan kredenal pengguna root atau AWS Identity and Access Management (IAM).
- Apakah permintaan dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi selengkapnya, lihat Elemen userIdentity CloudTrail .

AWS IoT Greengrass peristiwa data di CloudTrail

Peristiwa data memberikan informasi tentang operasi sumber daya yang dilakukan pada atau di sumber daya (misalnya, mendapatkan versi komponen atau konfigurasi penerapan). Ini juga dikenal sebagai operasi bidang data. Peristiwa data seringkali merupakan aktivitas volume tinggi. Secara default, CloudTrail tidak mencatat peristiwa data. Riwayat CloudTrail peristiwa tidak merekam peristiwa data.

Biaya tambahan berlaku untuk peristiwa data. Untuk informasi selengkapnya tentang CloudTrail harga, lihat AWS CloudTrail Harga.

Anda dapat mencatat peristiwa data untuk jenis AWS IoT Greengrass sumber daya menggunakan CloudTrail konsol AWS CLI, atau operasi CloudTrail API. Tabel di bagian ini menunjukkan jenis sumber daya yang tersedia untuk AWS IoT Greengrass.

- Untuk mencatat peristiwa data menggunakan CloudTrail konsol, buat penyimpanan data jejak atau peristiwa untuk mencatat peristiwa data, atau perbarui penyimpanan data jejak atau peristiwa yang ada untuk mencatat peristiwa data.

  1. Pilih Peristiwa data untuk mencatat peristiwa data.
  2. Dari daftar tipe peristiwa Data, pilih jenis sumber daya yang ingin Anda log peristiwa data.
  3. Pilih template pemilih log yang ingin Anda gunakan. Anda dapat mencatat semua peristiwa data untuk jenis sumber daya, mencatat semua `readOnly` peristiwa, mencatat semua `writeOnly` peristiwa, atau membuat templat pemilih log khusus untuk memfilter pada `readOnly`eventName, dan `resources.ARN` bidang.
- Untuk mencatat peristiwa data menggunakan AWS CLI, konfigurasikan `--advanced-event-selectors` parameter untuk mengatur `eventCategory` bidang sama dengan `Data` dan `resources.type` bidang sama dengan nilai tipe sumber daya (lihat tabel). Anda dapat menambahkan kondisi untuk memfilter nilai`readOnly`,`eventName`, dan `resources.ARN` bidang.
  - Untuk mengonfigurasi jejak untuk mencatat peristiwa data, jalankan put-event-selectorsperintah. Untuk informasi selengkapnya, lihat Mencatat peristiwa data untuk jejak dengan. AWS CLI
  - Untuk mengonfigurasi penyimpanan data peristiwa untuk mencatat peristiwa data, jalankan create-event-data-storeperintah untuk membuat penyimpanan data peristiwa baru untuk mencatat peristiwa data, atau jalankan update-event-data-storeperintah untuk memperbarui penyimpanan data peristiwa yang ada. Untuk informasi selengkapnya, lihat Mencatat peristiwa data untuk menyimpan data peristiwa dengan AWS CLI.

Tabel berikut mencantumkan jenis AWS IoT Greengrass sumber daya. Kolom tipe peristiwa data (konsol) menunjukkan nilai yang akan dipilih dari daftar tipe peristiwa Data di CloudTrail konsol. Kolom nilai resources.type menunjukkan **`resources.type`** nilai, yang akan Anda tentukan saat mengonfigurasi penyeleksi acara lanjutan menggunakan API atau. AWS CLI CloudTrail CloudTrailKolom API Data yang dicatat ke menampilkan panggilan API yang dicatat CloudTrail untuk jenis sumber daya.

Jenis peristiwa data (konsol)	nilai resources.type	API data masuk CloudTrail
Sertifikat IoT	`AWS::IoT::Certificate`	• VerifyClientDeviceIdentity • VerifyClientDeviceIoTCertificateAssociation
Versi komponen Greengrass IoT	`AWS::GreengrassV2::ComponentVersion`	• ResolveComponentCandidates
Penyebaran Greengrass IoT	`AWS::GreengrassV2::Deployment`	• GetDeploymentConfiguration
Hal IoT	`AWS::IoT::Thing`	• ListThingGroupsForCoreDevices • PutCertificateAuthorities • VerifyClientDeviceIoTCertificateAssociation

> **Note**
>
> Greengrass tidak mencatat peristiwa yang ditolak akses.

Anda dapat mengonfigurasi pemilih acara lanjutan untuk memfilter pada `eventName`,`readOnly`, dan `resources.ARN` bidang untuk mencatat hanya peristiwa yang penting bagi Anda.

Tambahkan filter `eventName` untuk menyertakan atau mengecualikan API data tertentu.

Untuk informasi lebih lanjut tentang bidang ini, lihat AdvancedFieldSelector.

Contoh berikut menunjukkan cara mengkonfigurasi penyeleksi lanjutan menggunakan. AWS CLI Ganti *TrailName*dan *wilayah* dengan informasi Anda sendiri.

Example — Log peristiwa data untuk hal-hal IoT

```
aws cloudtrail put-event-selectors --trail-name TrailName --region region \
--advanced-event-selectors \
'[
    {
            "Name": "Log all thing data events",
            "FieldSelectors": [
                { "Field": "eventCategory", "Equals": ["Data"] },
                { "Field": "resources.type", "Equals": ["AWS::IoT::Thing"] }
            ]
        }
]'
```

Example — Filter pada API IoT tertentu

```
aws cloudtrail put-event-selectors --trail-name TrailName --region region \
--advanced-event-selectors \
'[
    {
            "Name": "Log IoT Greengrass PutCertificateAuthorities API calls",
            "FieldSelectors": [
                { "Field": "eventCategory", "Equals": ["Data"] },
                { "Field": "resources.type", "Equals": ["AWS::IoT::Thing"] },
                { "Field": "eventName", "Equals": ["PutCertificateAuthorities"] }
            ]
    }
]'
```

Example — Log semua peristiwa data Greengrass

```
aws cloudtrail put-event-selectors --trail-name TrailName --region region \
--advanced-event-selectors \
'[
    {
        "Name": "Log all certificate data events",
        "FieldSelectors": [
            {
                "Field": "eventCategory",
                "Equals": [
```

```
                    “Data”
                ]
            },
            {
                “Field”: “resources.type”,
                “Equals”: [
                    “AWS::IoT::Certificate”
                ]
            }
        ]
    },
    {
        “Name”: “Log all component version data events”,
        “FieldSelectors”: [
            {
                “Field”: “eventCategory”,
                “Equals”: [
                    “Data”
                ]
            },
            {
                “Field”: “resources.type”,
                “Equals”: [
                    “AWS::GreengrassV2::ComponentVersion”
                ]
            }
        ]
    },
    {
        “Name”: “Log all deployment version”,
        “FieldSelectors”: [
            {
                “Field”: “eventCategory”,
                “Equals”: [
                    “Data”
                ]
            },
            {
                “Field”: “resources.type”,
                “Equals”: [
                    “AWS::GreengrassV2::Deployment”
                ]
            }
        ]
```

```
    },
    {
        “Name”: “Log all thing data events”,
        “FieldSelectors”: [
            {
                “Field”: “eventCategory”,
                “Equals”: [
                    “Data”
                ]
            },
            {
                “Field”: “resources.type”,
                “Equals”: [
                    “AWS::IoT::Thing”
                ]
            }
        ]
    }
]’
```

AWS IoT Greengrass acara manajemen di CloudTrail

Acara manajemen memberikan informasi tentang operasi manajemen yang dilakukan pada sumber daya di AWS akun Anda. Ini juga dikenal sebagai operasi pesawat kontrol. Secara default, CloudTrail mencatat peristiwa manajemen.

AWS IoT Greengrass mencatat semua operasi pesawat AWS IoT Greengrass kontrol sebagai peristiwa manajemen. Untuk daftar operasi bidang AWS IoT Greengrass kontrol yang AWS IoT Greengrass masuk ke log CloudTrail, lihat referensi AWS IoT Greengrass API, versi 2.

Memahami entri file AWS IoT Greengrass V2 log

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa menunjukkan satu permintaan dari sumber mana pun. Ini 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, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateDeployment` tindakan.

```
{
```

```
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Administrator",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Administrator"
    },
    "eventTime": "2021-01-06T02:38:05Z",
    "eventSource": "greengrass.amazonaws.com",
    "eventName": "CreateDeployment",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "aws-cli/2.1.9 Python/3.7.9 Windows/10 exe/AMD64 prompt/off command/
greengrassv2.create-deployment",
    "requestParameters": {
        "deploymentPolicies": {
            "failureHandlingPolicy": "DO_NOTHING",
            "componentUpdatePolicy": {
                "timeoutInSeconds": 60,
                "action": "NOTIFY_COMPONENTS"
            },
            "configurationValidationPolicy": {
                "timeoutInSeconds": 60
            }
        },
        "deploymentName": "Deployment for MyGreengrassCoreGroup",
        "components": {
            "aws.greengrass.Cli": {
                "componentVersion": "2.0.3"
            }
        },
        "iotJobConfiguration": {},
        "targetArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
MyGreengrassCoreGroup"
    },
    "responseElements": {
        "iotJobArn": "arn:aws:iot:us-west-2:123456789012:job/fdfeba1d-ac6d-44ef-
ab28-54f684ea578d",
        "iotJobId": "fdfeba1d-ac6d-44ef-ab28-54f684ea578d",
        "deploymentId": "4196dddc-0a21-4c54-a985-66a525f6946e"
    },
    "requestID": "311b9529-4aad-42ac-8408-c06c6fec79a9",
```

```
    "eventID": "c0f3aa2c-af22-48c1-8161-bad4a2ab1841",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123456789012"
}
```

Kumpulkan data telemetri kondisi sistem dari perangkat inti AWS IoT Greengrass

Data telemetri kesehatan sistem adalah data diagnostik yang dapat membantu Anda memantau kinerja operasi kritis pada perangkat inti Greengrass Anda. Anda dapat membuat proyek dan aplikasi untuk mengambil, menganalisis, mengubah, dan melaporkan data telemetri dari perangkat edge Anda. Pakar domain, seperti para insinyur proses, dapat menggunakan aplikasi ini untuk mendapatkan wawasan tentang kesehatan armada.

Anda dapat menggunakan metode berikut untuk mengumpulkan data telemetri dari perangkat inti Greengrass Anda:

- Komponen pemancar telemetri nukleus —Komponen pemancar telemetri nukleus (`aws.greengrass.telemetry.NucleusEmitter`) pada perangkat inti Greengrass menerbitkan data telemetri ke topik secara default. `$local/greengrass/telemetry` Anda dapat menggunakan data yang dipublikasikan ke topik ini untuk bertindak secara lokal di perangkat inti Anda, bahkan ketika perangkat Anda memiliki konektivitas terbatas ke cloud. Secara opsional, Anda juga dapat mengonfigurasi komponen untuk mempublikasikan data telemetri ke topik AWS IoT Core MQTT pilihan Anda.

  Anda harus menyebarkan komponen pemancar nukleus ke perangkat inti untuk mempublikasikan data telemetri. Tidak ada biaya yang terkait dengan penerbitan data telemetri ke topik lokal. Namun, penggunaan topik MQTT untuk mempublikasikan data ke tunduk pada AWS Cloud harga. AWS IoT Core

  AWS IoT Greengrassmenyediakan beberapa komponen komunitas untuk membantu Anda menganalisis dan memvisualisasikan data telemetri secara lokal di perangkat inti Anda menggunakan InfluxDB dan Grafana. Komponen-komponen ini menggunakan data telemetri dari komponen pemancar nukleus. Untuk informasi selengkapnya, lihat README untuk komponen penerbit InfluxDB.

- Agen telemetri —Agen telemetri pada perangkat inti Greengrass mengumpulkan data telemetri lokal dan menerbitkannya ke Amazon tanpa memerlukan interaksi pelanggan. EventBridge Perangkat inti mempublikasikan data telemetri dengan upaya terbaik. EventBridge Sebagai contoh, perangkat inti mungkin gagal untuk mengirimkan data telemetri saat offline.

  Fitur agen telemetri diaktifkan secara default untuk semua perangkat inti Greengrass. Anda secara otomatis mulai menerima data segera setelah Anda mengatur perangkat inti Greengrass. Selain biaya tautan data Anda, transfer data dari perangkat inti ke AWS IoT Core tanpa biaya. Hal ini karena agen tersebut menerbitkan ke topik tersimpan AWS. Namun, tergantung pada kasus penggunaan Anda, Anda dapat dikenakan biaya saat menerima atau memproses data.

  > **Note**
  >
  > Amazon EventBridge adalah layanan bus acara yang dapat Anda gunakan untuk menghubungkan aplikasi Anda dengan data dari berbagai sumber, seperti perangkat inti Greengrass. Untuk informasi lebih lanjut, lihat Apa itu Amazon EventBridge? di Panduan EventBridge Pengguna Amazon.

Untuk memastikan bahwa perangkat lunak AWS IoT Greengrass Core berfungsi dengan baik, AWS IoT Greengrass gunakan data untuk tujuan pengembangan dan peningkatan kualitas. Fitur ini juga membantu menginformasikan kemampuan edge yang baru dan ditingkatkan. AWS IoT Greengrassmenyimpan data telemetri hingga tujuh hari.

Bagian ini menjelaskan cara mengkonfigurasi dan menggunakan agen telemetri. Untuk informasi tentang mengonfigurasi komponen pemancar telemetri nukleus, lihat. Pemancar telemetri nukleus

Topik

Metrik telemetri

Tabel berikut menjelaskan metrik yang diterbitkan oleh agen telemetri.

Nama	Penjelasan
Sistem	
`SystemMemUsage`	Jumlah memori yang saat ini digunakan oleh semua aplikasi pada perangkat inti Greengrass, termasuk sistem operasi.
`CpuUsage`	Jumlah CPU yang saat ini digunakan oleh semua aplikasi pada perangkat inti Greengrass, termasuk sistem operasi.
`TotalNumberOfFDs`	Bilangan deskriptor file yang disimpan oleh sistem operasi perangkat inti Greengrass. Satu file deskriptor secara unik mengidentifikasi satu file yang terbuka.
Inti Greengrass	
`NumberOfComponentsRunning`	Jumlah komponen yang berjalan pada perangkat inti Greengrass.
`NumberOfComponentsErrored`	Jumlah komponen yang berada dalam keadaan kesalahan pada perangkat inti Greengrass.
`NumberOfComponentsInstalled`	Jumlah komponen yang diinstal pada perangkat inti Greengrass.

Nama	Penjelasan	
`NumberOfComponents Starting`	Jumlah komponen yang dimulai pada perangkat inti Greengrass.	
`NumberOfComponents New`	Jumlah komponen yang baru pada perangkat inti Greengrass.	
`NumberOfComponents Stopping`	Jumlah komponen yang berhenti pada perangkat inti Greengrass.	
`NumberOfComponents Finished`	Jumlah komponen yang diselesaikan pada perangkat inti Greengrass.	
`NumberOfComponents Broken`	Jumlah komponen yang rusak pada perangkat inti Greengrass.	
`NumberOfComponents Stateless`	Jumlah komponen yang stateless pada perangkat inti Greengrass.	
Autentikasi perangkat klien - Fitur ini memerlukan v2.4.0 atau yang lebih baru dari komponen autentikasi perangkat klien.		
`VerifyClientDevice Identity.Success`	Berapa kali memverifikasi bahwa identitas perangkat klien berhasil.	
`VerifyClientDevice Identity.Failure`	Berapa kali memverifikasi bahwa identitas perangkat klien gagal.	

Nama	Penjelasan	
`AuthorizeClientDeviceActions.Success`	Berapa kali perangkat klien diizinkan untuk menyelesaikan tindakan yang diminta.	
`AuthorizeClientDeviceActions.Failure`	Berapa kali perangkat klien tidak diizinkan untuk menyelesaikan tindakan yang diminta.	
`GetClientDeviceAuthToken.Success`	Berapa kali perangkat klien berhasil diautentikasi.	
`GetClientDeviceAuthToken.Failure`	Berapa kali perangkat klien tidak dapat diautentikasi.	
`SubscribeToCertificateUpdates.Success`	Jumlah langganan yang berhasil untuk pembaruan sertifikat.	
`SubscribeToCertificateUpdates.Failure`	Jumlah upaya yang gagal untuk berlangganan pembaruan sertifikat.	
`ServiceError`	Jumlah kesalahan internal yang tidak tertangani di seluruh autentikasi perangkat klien.	
Stream manager — Fitur ini membutuhkan v2.7.0 atau yang lebih baru dari komponen inti Greengrass.		
`BytesAppended`	Jumlah byte data ditambahkan ke pengelola pengaliran.	

Nama	Penjelasan
`BytesUploadedToIoTAnalytics`	Jumlah byte data yang diekspor pengelola pengaliran ke saluran di AWS IoT Analytics.
`BytesUploadedToKinesis`	Jumlah byte data yang diekspor pengelola pengaliran ke pengaliran di Amazon Kinesis Data Streams.
`BytesUploadedToIoTSiteWise`	Jumlah byte data yang diekspor pengelola pengaliran ke properti aset di AWS IoT SiteWise.
`BytesUploadedToS3`	Jumlah byte data yang diekspor pengelola pengaliran ke objek di Amazon S3.

Konfigurasikan pengaturan agen telemetri

Agen telemetri menggunakan pengaturan default berikut:

- Agen telemetri mengumpulkan data telemetri setiap jam.
- Agen telemetri menerbitkan pesan telemetri setiap 24 jam.

Agen telemetri menerbitkan data menggunakan protokol MQTT dengan tingkat kualitas layanan (QoS) 0, yang berarti bahwa ia 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 inti ke AWS IoT Core tidak dipungut biaya. Hal ini karena agen tersebut menerbitkan ke topik tersimpan AWS. Namun, tergantung pada kasus penggunaan Anda, Anda dapat dikenakan biaya saat menerima atau memproses data.

Anda dapat mengaktifkan atau menonaktifkan fitur agen telemetri untuk setiap perangkat inti Greengrass. Anda juga dapat mengonfigurasi interval di mana perangkat inti mengagregasi dan menerbitkan data. Untuk mengonfigurasi telemetri, sesuaikan parameter konfigurasi telemetri saat Anda men-deploy komponen inti Greengrass.

Berlangganan data telemetri di EventBridge

Anda dapat membuat aturan di Amazon EventBridge yang menentukan cara memproses data telemetri yang diterbitkan dari agen telemetri di perangkat inti Greengrass. Saat EventBridge menerima data, itu memanggil tindakan target yang ditentukan dalam aturan Anda. Misalnya, Anda dapat membuat aturan acara yang mengirim notifikasi, menyimpan informasi peristiwa, mengambil tindakan korektif, atau memanggil peristiwa lain.

Peristiwa telemetri

Peristiwa telemetri menggunakan format berikut.


```
{
  "version": "0",
  "id": "a09d303e-2f6e-3d3c-a693-8e33f4fe3955",
  "detail-type": "Greengrass Telemetry Data",
  "source": "aws.greengrass",
  "account": "123456789012",
  "time": "2020-11-30T20:45:53Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "ThingName": "MyGreengrassCore",
    "Schema": "2020-07-30",
    "ADP": [
      {
        "TS": 1602186483234,
        "NS": "SystemMetrics",
        "M": [
          {
            "N": "TotalNumberOfFDs",
            "Sum": 6447.0,
            "U": "Count"
          },
          {
            "N": "CpuUsage",
            "Sum": 15.458333333333332,
```

```
            "U": "Percent"
          },
          {
            "N": "SystemMemUsage",
            "Sum": 10201.0,
            "U": "Megabytes"
          }
        ]
      },
      {
        "TS": 1602186483234,
        "NS": "GreengrassComponents",
        "M": [
          {
            "N": "NumberOfComponentsStopping",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsStarting",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsBroken",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsFinished",
            "Sum": 1.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsInstalled",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsRunning",
            "Sum": 7.0,
            "U": "Count"
          },
          {
```

```
            "N": "NumberOfComponentsNew",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsErrored",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsStateless",
            "Sum": 0.0,
            "U": "Count"
          }
        ]
      },
      {
        "TS": 1602186483234,
        "NS": "aws.greengrass.ClientDeviceAuth",
        "M": [
          {
            "N": "VerifyClientDeviceIdentity.Success",
            "Sum": 3.0,
            "U": "Count"
          },
          {
            "N": "VerifyClientDeviceIdentity.Failure",
            "Sum": 1.0,
            "U": "Count"
          },
          {
            "N": "AuthorizeClientDeviceActions.Success",
            "Sum": 20.0,
            "U": "Count"
          },
          {
            "N": "AuthorizeClientDeviceActions.Failure",
            "Sum": 5.0,
            "U": "Count"
          },
          {
            "N": "GetClientDeviceAuthToken.Success",
            "Sum": 5.0,
            "U": "Count"
```

```
          },
          {
            "N": "GetClientDeviceAuthToken.Failure",
            "Sum": 2.0,
            "U": "Count"
          },
          {
            "N": "SubscribeToCertificateUpdates.Success",
            "Sum": 10.0,
            "U": "Count"
          },
          {
            "N": "SubscribeToCertificateUpdates.Failure",
            "Sum": 1.0,
            "U": "Count"
          },
          {
            "N": "ServiceError",
            "Sum": 3.0,
            "U": "Count"
          }
        ]
      },
      {
        "TS": 1602186483234,
        "NS": "aws.greengrass.StreamManager",
        "M": [
          {
            "N": "BytesAppended",
            "Sum": 157745524.0,
            "U": "Bytes"
          },
          {
            "N": "BytesUploadedToIoTAnalytics",
            "Sum": 149012.0,
            "U": "Bytes"
          },
          {
            "N": "BytesUploadedToKinesis",
            "Sum": 12192.0,
            "U": "Bytes"
          },
          {
            "N": "BytesUploadedToIoTSiteWise",
```

```
            "Sum": 13321.0,
            "U": "Bytes"
          },
          {
            "N": "BytesUploadedToS3",
            "Sum": 12213.0,
            "U": "Bytes"
          }
        ]
      }
    ]
  }
}
```

Rangkaian ADP berisi daftar titik data agregat yang memiliki properti sebagai berikut:

`TS`

Stempel waktu kapan data dikumpulkan.

`NS`

Namespace metrik.

`M`

Daftar metrik. Metrik ini berisi properti berikut:

`N`

Nama metrik.

`Sum`

Jumlah nilai metrik dalam peristiwa telemetri ini.

`U`

Unit nilai metrik.

Untuk informasi lebih lanjut tentang setiap metrik, lihatMetrik telemetri.

Prasyarat untuk membuat aturan EventBridge

Sebelum Anda membuat EventBridge aturan untukAWS IoT Greengrass, Anda harus melakukan hal berikut:

- Biasakan diri Anda dengan acara, aturan, dan target di EventBridge.
- Buat dan konfigurasikan 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 di Wilayah AWS tempat Anda membuat sumber daya Greengrass Anda. Untuk informasi selengkapnya, lihat Titik akhir layanan dan kuota di. Referensi Umum AWS

Untuk informasi lebih lanjut, lihat Apa itu Amazon EventBridge? dan Memulai Amazon EventBridge di Panduan EventBridge Pengguna Amazon.

Buat aturan peristiwa untuk mendapatkan data telemetri (konsol)

Gunakan langkah-langkah berikut untuk menggunakan aturan AWS Management Console untuk membuat EventBridge aturan yang menerima data telemetri yang diterbitkan oleh perangkat inti 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 dari AWS 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. Di bawah Tentukan pola, konfigurasikan pola aturan.
   a. Pilih Pola kejadian.
   b. Pilih Pola yang ditentukan sebelumnya oleh layanan.
   c. Untuk Penyedia layanan, pilih AWS.
   d. Untuk Nama layanan, pilih Greengrass.
   e. Untuk Jenis peristiwa, pilih Data Telemetri Greengrass.
4. Di bawah Pilih bus peristiwa, jaga default opsi bus peristiwa.
5. Di bawah Pilih target, konfigurasikan target Anda. Contoh berikut menggunakan antrean Amazon SQS, tetapi Anda dapat mengonfigurasi jenis target lainnya.
   a. Untuk Target, pilih Antrean SQS.
   b. Untuk Antrean, pilih antrean target Anda.
6. Di bawah Tag - opsional, tentukan tag untuk aturan tersebut atau biarkan kolom tersebut kosong.

7. Pilih Buat.

Buat aturan peristiwa untuk mendapatkan data telemetri (CLI)

Gunakan langkah-langkah berikut untuk menggunakan aturan AWS CLI untuk membuat EventBridge aturan yang menerima data telemetri yang diterbitkan oleh perangkat inti Greengrass. Hal ini memungkinkan server web, alamat email, dan pelanggan topik lainnya untuk menanggapi peristiwa tersebut.

1. Buat aturan.

   - Ganti *thing-name* dengan nama objek perangkat inti.

   Linux or Unix

   ```
   aws events put-rule \
     --name MyGreengrassTelemetryEventRule \
     --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail\": {\"ThingName
   \": [\"thing-name\"]}}"
   ```

   Windows Command Prompt (CMD)

   ```
   aws events put-rule ^
     --name MyGreengrassTelemetryEventRule ^
     --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail\": {\"ThingName
   \": [\"thing-name\"]}}"
   ```

   PowerShell

   ```
   aws events put-rule `
     --name MyGreengrassTelemetryEventRule `
     --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail\": {\"ThingName
   \": [\"thing-name\"]}}"
   ```

   Properti yang dihilangkan dari tersebut 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.

Linux or Unix

```
aws events put-targets \
  --rule MyGreengrassTelemetryEventRule \
  --targets "Id"="1","Arn"="queue-arn"
```

Windows Command Prompt (CMD)

```
aws events put-targets ^
  --rule MyGreengrassTelemetryEventRule ^
  --targets "Id"="1","Arn"="queue-arn"
```

PowerShell

```
aws events put-targets `
  --rule MyGreengrassTelemetryEventRule `
  --targets "Id"="1","Arn"="queue-arn"
```

> **Note**
>
> Untuk mengizinkan Amazon EventBridge memanggil antrean target, Anda harus menambahkan kebijakan berbasis sumber daya ke topik Anda. Untuk informasi selengkapnya, lihat izin Amazon SQS di Panduan Pengguna Amazon EventBridge .

Untuk informasi selengkapnya, lihat Peristiwa dan pola peristiwa EventBridge di Panduan EventBridge Pengguna Amazon.

Dapatkan pemberitahuan status kesehatan penerapan dan komponen

Aturan EventBridge acara Amazon memberi Anda pemberitahuan tentang perubahan status untuk penerapan Greengrass yang diterima oleh perangkat Anda dan untuk komponen yang diinstal pada perangkat Anda. EventBridge memberikan aliran peristiwa sistem yang mendekati waktu nyata

yang menjelaskan perubahan AWS sumber daya. AWS IoT Greengrass mengirimkan acara ini ke EventBridge atas dasar upaya terbaik. Ini berarti bahwa AWS IoT Greengrass upaya untuk mengirim semua acara ke EventBridge tetapi, dalam beberapa kasus yang jarang terjadi, suatu peristiwa mungkin tidak disampaikan. Selain itu, AWS IoT Greengrass mungkin mengirim beberapa salinan dari peristiwa tertentu, yang berarti bahwa pendengar acara Anda mungkin tidak menerima peristiwa dalam urutan kejadian terjadi.

> **Note**
>
> Amazon EventBridge adalah layanan bus acara yang dapat Anda gunakan untuk menghubungkan aplikasi Anda dengan data dari berbagai sumber, seperti perangkat inti Greengrass dan penyebaran serta pemberitahuan komponen. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge? di Panduan EventBridge Pengguna Amazon.

Topik

Acara perubahan status penerapan

AWS IoT Greengrass memancarkan peristiwa ketika penerapan memasuki status berikut:`FAILED`,,, `SUCCEEDED` `COMPLETEDREJECTED`, dan. `CANCELED` Anda dapat membuat EventBridge aturan yang berjalan untuk semua transisi status atau transisi ke status yang Anda tentukan. Saat penerapan memasuki status yang memulai aturan, EventBridge memanggil tindakan target yang ditentukan 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.

peristiwa untuk perubahan keadaan deployment menggunakan format berikut:

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Effective Deployment Status Change",
    "source":"aws.greengrass",
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
    "detail":{
        "deploymentId": "4f38f1a7-3dd0-42a1-af48-EXAMPLE09681",
        "coreDeviceExecutionStatus": "FAILED|SUCCEEDED|COMPLETED|REJECTED|CANCELED",
        "statusDetails": {
          "errorStack": ["DEPLOYMENT_FAILURE", "ARTIFACT_DOWNLOAD_ERROR", "S3_ERROR",
 "S3_ACCESS_DENIED", "S3_HEAD_OBJECT_ACCESS_DENIED"],
          "errorTypes": ["DEPENDENCY_ERROR", "PERMISSION_ERROR"],
        },
        "reason": "S3_HEAD_OBJECT_ACCESS_DENIED: FAILED_NO_STATE_CHANGE: Failed to
 download artifact name: 's3://pentest27/nucleus/281/aws.greengrass.nucleus.zip' for
 component aws.greengrass.Nucleus-2.8.1, reason: S3 HeadObject returns 403 Access
 Denied. Ensure the IAM role associated with the core device has a policy granting
 s3:GetObject. null (Service: S3, Status Code: 403, Request ID: HR94ZNT2161DAR58,
 Extended Request ID: wTX4DDI+qigQt3uzwl9rlnQiYlBgwvPm/KJFWeFAn9t1mnGXTms/
luLCYANgq08RIH+x2H+hEKc=)"
     }
}
```

Anda dapat membuat aturan dan acara yang akan memperbarui Anda tentang status penerapan. Sebuah peristiwa dimulai ketika penerapan selesai sebagai salah satu`FAILED`,,, `SUCCEEDEDCOMPLETED`, `REJECTED` atau. `CANCELED` Jika penerapan gagal pada perangkat inti, Anda akan menerima respons terperinci yang menjelaskan mengapa penerapan gagal. Untuk informasi selengkapnya tentang kode kesalahan penerapan, lihatKode kesalahan penyebaran terperinci.

Status penyebaran

- `FAILED`. Deployment gagal.

- `SUCCEEDED`. Penyebaran yang ditargetkan ke grup sesuatu berhasil diselesaikan.
- `COMPLETED`. Penyebaran yang ditargetkan untuk sesuatu yang berhasil diselesaikan.
- `REJECTED`. Penyebaran ditolak. Untuk informasi lebih lanjut, lihat `statusDetails` bidangnya.
- `CANCELED`. Penerapan dibatalkan oleh pengguna.

Ada kemungkinan bahwa peristiwa dapat diduplikasi atau rusak. Untuk menentukan urutan peristiwa, gunakan properti `time` ini.

Untuk daftar lengkap kode kesalahan di `errorStacks` dan`errorTypes`, lihat Kode kesalahan penyebaran terperinci dan Kode status komponen rinci.

Acara perubahan status komponen

Untuk AWS IoT Greengrass versi 2.12.2 dan versi sebelumnya, Greengrass memancarkan peristiwa saat komponen memasuki status berikut: dan. `ERRORED` `BROKEN` Untuk Greengrass nucleus versi 2.12.3 dan yang lebih baru, Greengrass memancarkan peristiwa ketika komponen memasuki status berikut:,,, dan. `ERRORED` `BROKEN` `RUNNING` `FINISHED` Greengrass juga akan memancarkan peristiwa saat penerapan selesai. Anda dapat membuat EventBridge aturan yang berjalan untuk semua transisi status atau transisi ke status yang Anda tentukan. Ketika komponen yang diinstal memasuki status yang memulai aturan, EventBridge memanggil tindakan target yang ditentukan dalam aturan. Hal ini mengizinkan Anda untuk mengirim pemberitahuan, menangkap informasi peristiwa, mengambil tindakan korektif, atau menginisiasi peristiwa lain untuk merespon perubahan keadaan.

Acara untuk perubahan status komponen menggunakan format berikut:

Greengrass nucleus v2.12.2 and earlier

<title>Status komponen: ERRORED atau BROKEN</title>

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Installed Component Status Change",
    "source":"aws.greengrass",
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
```

```
    "detail": {
       "components": [
          {
            "componentName": "MyComponent",
            "componentVersion": "1.0.0",
            "root": true,
            "lifecycleState": "ERRORED|BROKEN",
            "lifecycleStatusCodes": ["STARTUP_ERROR"],
            "lifecycleStateDetails": "An error occurred during startup. The startup
 script exited with code 1."
          }
        ]
    }
}
```

Greengrass nucleus v2.12.3 and later

<title>Status komponen: ERRORED atau BROKEN</title>

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Installed Component Status Change",
    "source":"aws.greengrass",
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
    "detail": {
       "components": [
          {
            "componentName": "MyComponent",
            "componentVersion": "1.0.0",
            "root": true,
            "lifecycleState": "ERRORED|BROKEN",
            "lifecycleStatusCodes": ["STARTUP_ERROR"],
            "lifecycleStateDetails": "An error occurred during startup. The startup
 script exited with code 1."
          }
        ]
    }
}
```

<title>Status komponen: RUNNING atau FINISHED</title>

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Installed Component Status Change",
    "source":"aws.greengrass",
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
    "detail": {
       "components": [
          {
            "componentName": "MyComponent",
            "componentVersion": "1.0.0",
            "root": true,
            "lifecycleState": "RUNNING|FINISHED",
            "lifecycleStateDetails": null
          }
        ]
    }
}
```

Anda dapat membuat aturan dan acara yang akan memperbarui Anda tentang status komponen yang diinstal. Peristiwa dimulai saat komponen mengubah status pada perangkat. Anda akan menerima tanggapan terperinci yang menjelaskan mengapa suatu komponen salah atau rusak. Anda juga akan menerima kode status yang akan menunjukkan alasan kegagalan tersebut. Untuk informasi selengkapnya tentang kode status komponen, lihatKode status komponen rinci.

Prasyarat untuk membuat aturan EventBridge

Sebelum Anda membuat EventBridge aturan untuk AWS IoT Greengrass, lakukan hal berikut:

- Biasakan diri Anda dengan acara, aturan, dan target di EventBridge.
- Buat dan konfigurasikan target yang dipanggil oleh EventBridge aturan Anda. Aturan dapat menginvokasi 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, lihat Apa itu Amazon EventBridge? dan Memulai dengan Amazon EventBridge di Panduan EventBridge Pengguna Amazon.

Konfigurasikan pemberitahuan kesehatan perangkat (konsol)

Gunakan langkah-langkah berikut untuk membuat EventBridge aturan yang menerbitkan topik Amazon SNS saat status penerapan berubah untuk grup. 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 dari AWS sumber daya di Panduan EventBridge Pengguna Amazon.

1. Buka EventBridgekonsol Amazon.
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. Untuk bus acara, pilih bus acara yang ingin Anda kaitkan dengan aturan ini. Jika Anda ingin aturan ini cocok dengan acara yang berasal dari akun Anda, pilih bus acara AWS default. Ketika AWS layanan di akun Anda memancarkan suatu acara, itu selalu masuk ke bus acara default akun Anda.
6. Untuk Tipe aturan, pilih Aturan dengan pola peristiwa.
7. Pilih Selanjutnya.
8. Untuk sumber Acara, pilih AWS acara.
9. Untuk pola Acara, pilih AWS layanan.
10. Untuk AWS layanan, pilih Greengrass.
11. Untuk jenis Acara, pilih dari berikut ini:

    - Untuk peristiwa penerapan, pilih Greengrass V2 Perubahan Status Penerapan Efektif.
    - Untuk event komponen, pilih Greengrass V2 Installed Component Status Change.

12. Pilih Selanjutnya.
13. Untuk Jenis target, pilih Layanan AWS .
14. Untuk Pilih 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.
15. Pilih Selanjutnya.
16. Tinjau detail aturan dan pilih Buat aturan.

Konfigurasikan pemberitahuan kesehatan perangkat (CLI)

Gunakan langkah-langkah berikut untuk membuat EventBridge aturan yang menerbitkan topik Amazon SNS saat ada peristiwa perubahan status Greengrass. Hal ini memungkinkan server web, alamat email, dan pelanggan topik lainnya untuk menanggapi peristiwa tersebut.

1. Buat aturan.

   - Untuk peristiwa perubahan status penerapan.

     ```
     aws events put-rule \
       --name TestRule \
       --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail-type\":
      [\"Greengrass V2 Effective Deployment Status Change\"]}"
     ```

   - Untuk peristiwa perubahan status komponen.

     ```
     aws events put-rule \
       --name TestRule \
       --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail-type\":
      [\"Greengrass V2 Installed Component Status Change\"]}"
     ```

   Properti yang dihilangkan dari tersebut 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 memanggil topik target Anda, Anda harus menambahkan kebijakan berbasis sumber daya ke topik Anda. Untuk informasi selengkapnya, lihat izin Amazon SNS di Panduan Pengguna Amazon EventBridge .

Untuk informasi selengkapnya, lihat Peristiwa dan pola acara EventBridge di Panduan EventBridge Pengguna Amazon.

Konfigurasikan pemberitahuan kesehatan perangkat (AWS CloudFormation)

Gunakan AWS CloudFormation templat untuk membuat EventBridge aturan yang mengirim pemberitahuan tentang perubahan status untuk penerapan grup Greengrass Anda. Untuk informasi selengkapnya, lihat referensi jenis EventBridge sumber daya Amazon di Panduan AWS CloudFormation Pengguna.

Lihat juga

- Periksa status deployment perangkat
- Apa itu Amazon EventBridge? di Panduan EventBridge Pengguna Amazon

Periksa status perangkat inti Greengrass

Perangkat inti Greengrass melaporkan status komponen perangkat lunak mereka ke. AWS IoT Greengrass Anda dapat memeriksa ringkasan kesehatan setiap perangkat, dan Anda dapat memeriksa status masing-masing komponen di setiap perangkat.

Perangkat inti memiliki status kesehatan berikut:

- `HEALTHY`— Perangkat lunak AWS IoT Greengrass Core dan semua komponen berjalan tanpa masalah pada perangkat inti.
- `UNHEALTHY`— Perangkat lunak AWS IoT Greengrass Inti atau komponen berada dalam keadaan kesalahan pada perangkat inti.

> **Note**
>
> AWS IoT Greengrass bergantung pada perangkat individual untuk mengirim pembaruan status ke file. AWS Cloud Jika perangkat lunak AWS IoT Greengrass Core tidak berjalan di perangkat, atau jika perangkat tidak terhubung ke perangkat AWS Cloud, maka status perangkat yang dilaporkan mungkin tidak mencerminkan statusnya saat ini. Cap waktu status menunjukkan kapan status perangkat terakhir diperbarui.
> Perangkat inti mengirim pembaruan status pada waktu-waktu berikut:
>
> - Saat perangkat lunak AWS IoT Greengrass inti dimulai
> - Ketika perangkat inti menerima penerapan dari AWS Cloud
> - Untuk Greengrass nucleus 2.12.2 dan sebelumnya, perangkat inti mengirimkan pembaruan status saat status komponen apa pun pada perangkat inti menjadi atau `ERRORED` `BROKEN`
> - Untuk Greengrass nucleus 2.12.3 dan yang lebih baru, perangkat inti mengirimkan pembaruan status saat status komponen apa pun pada perangkat inti menjadi,,, atau `ERRORED` `BROKEN` `RUNNING` `FINISHED`
> - Pada interval reguler yang dapat Anda konfigurasikan, yang defaultnya 24 jam
>
> Untuk AWS IoT Greengrass Core v2.7.0 dan yang lebih baru, perangkat inti mengirimkan pembaruan status saat penerapan lokal dan penyebaran cloud terjadi

Topik

Periksa kesehatan perangkat inti

Anda dapat memeriksa status perangkat inti individu.

Untuk memeriksa status perangkat inti (AWS CLI)

- Jalankan perintah berikut untuk mengambil status deployment. Ganti *coreDeviceName* dengan nama perangkat inti untuk kueri.

  ```
  aws greengrassv2 get-core-device --core-device-thing-name coreDeviceName
  ```

  Tanggapan berisi informasi tentang perangkat inti, termasuk statusnya.

Periksa kesehatan grup perangkat inti

Anda dapat memeriksa status grup perangkat inti (grup objek).

Untuk memeriksa status grup perangkat (AWS CLI)

- Jalankan perintah berikut untuk mengambil status beberapa perangkat inti. Ganti ARN dalam perintah tersebut dengan ARN dari grup objek untuk kueri.

  ```
  aws greengrassv2 list-core-devices --thing-group-arn "arn:aws:iot:region:account-
  id:thinggroup/thingGroupName"
  ```

  Tanggapan berisi daftar perangkat inti dalam grup objek. Setiap entri dalam daftar berisi status perangkat inti.

Periksa status komponen perangkat inti

Anda dapat memeriksa status, seperti keadaan siklus hidup, dari komponen perangkat lunak pada perangkat inti. Untuk informasi lebih lanjut tentang keadaan siklus hidup komponen, lihat Kembangkan AWS IoT Greengrass komponen.

Untuk memeriksa status komponen pada perangkat inti (AWS CLI)

- Jalankan perintah berikut untuk mengambil status komponen pada perangkat inti. Ganti *coreDeviceName* dengan nama perangkat inti untuk kueri.

```
aws greengrassv2 list-installed-components --core-device-thing-name coreDeviceName
```

Tanggapan berisi daftar komponen yang berjalan pada perangkat inti. Setiap entri dalam daftar berisi status siklus hidup komponen, termasuk seberapa terkini status data dan kapan perangkat inti Greengrass terakhir mengirim pesan yang berisi komponen tertentu ke cloud. Respons juga akan mencakup sumber penyebaran terbaru yang membawa komponen ke perangkat inti Greengrass.

> **Note**
>
> Perintah ini mengambil daftar komponen yang dipaginasi yang dijalankan oleh perangkat inti Greengrass. Secara default, daftar ini tidak menyertakan komponen yang digunakan sebagai dependensi komponen lain. Anda dapat menyertakan dependensi dalam respons dengan menyetel `topologyFilter` parameter ke. `ALL`

Jalankan fungsi AWS Lambda

> **Note**
>
> AWS IoT Greengrasssaat ini tidak mendukung fitur ini di perangkat inti Windows.

Anda dapat mengimpor fungsi AWS Lambda sebagai komponen yang berjalan pada perangkat inti AWS IoT Greengrass. Anda mungkin ingin melakukan hal ini dalam kasus berikut:

- Anda memiliki kode aplikasi dalam fungsi Lambda yang ingin Anda deploy ke perangkat inti.
- Anda memiliki aplikasi V1 AWS IoT Greengrass yang ingin Anda jalankan di perangkat inti AWS IoT Greengrass V2. Untuk informasi selengkapnya, lihat Langkah 2: Membuat dan menyebarkan AWS IoT Greengrass V2 komponen untuk memigrasi aplikasi AWS IoT Greengrass V1.

fungsi Lambda meliputi dependensi pada komponen-komponen berikut. Anda tidak perlu mendefinisikan komponen ini sebagai dependensi ketika Anda mengimpor fungsi tersebut. Ketika Anda men-deploy komponen fungsi Lambda, deployment tersebut mencakup dependensi komponen Lambda ini.

- Komponen peluncur Lambda (`aws.greengrass.LambdaLauncher`) menangani proses dan konfigurasi lingkungan.
- Komponen manajer Lambda (`aws.greengrass.LambdaManager`) menangani komunikasi antar proses dan penskalaan.
- Komponen waktu aktif Lambda (`aws.greengrass.LambdaRuntimes`) menyediakan artefak untuk setiap waktu aktif Lambda yang didukung.

Topik

Persyaratan

Perangkat inti dan fungsi Lambda Anda harus memenuhi persyaratan berikut bagi Anda untuk menjalankan fungsi pada perangkat lunak inti AWS IoT Greengrass:

- Perangkat inti Anda harus memenuhi persyaratan untuk menjalankan fungsi Lambda. Jika Anda ingin perangkat inti untuk menjalankan fungsi Lambda kontainer, perangkat harus memenuhi persyaratan untuk melakukannya. Untuk informasi selengkapnya, lihat Persyaratan fungsi Lambda.
- Anda harus menginstal bahasa pemrograman yang digunakan oleh fungsi Lambda pada perangkat inti Anda.

> **Tip**
>
> Anda dapat membuat komponen yang menginstal bahasa pemrograman, dan kemudian menentukan komponen itu sebagai dependensi dari komponen fungsi Lambda Anda. Greengrass mendukung semua versi runtime Python, Node.js, dan Java yang didukung Lambda. Greengrass tidak menerapkan batasan tambahan apa pun pada versi runtime Lambda yang tidak digunakan lagi. Anda dapat menjalankan fungsi Lambda yang menggunakan runtime usang iniAWS IoT Greengrass, tetapi Anda tidak dapat membuatnya. AWS Lambda Untuk informasi lebih lanjut tentang dukungan AWS IoT Greengrass untuk waktu aktif Lambda, lihat Jalankan fungsi AWS Lambda.

Konfigurasikan siklus hidup fungsi Lambda

Siklus hidup fungsi Greengrass Lambda menentukan kapan fungsi dimulai dan bagaimana menciptakan dan menggunakan kontainer. Siklus hidup juga menentukan bagaimana perangkat lunak AWS IoT Greengrass Core mempertahankan variabel dan logika preprocessing yang berada di luar fungsi handler.

AWS IoT Greengrassmendukung on-demand (default) dan siklus hidup yang berumur panjang:

- Fungsi sesuai permintaan dimulai saat dipanggil dan berhenti ketika tidak ada tugas yang tersisa untuk dijalankan. Setiap pemanggilan fungsi membuat wadah terpisah, juga disebut kotak pasir, untuk memproses pemanggilan, kecuali wadah yang ada tersedia untuk digunakan kembali. Kontainer mana pun dapat memproses data yang Anda kirim ke fungsi.

  Beberapa pemanggilan fungsi on-demand dapat berjalan secara bersamaan.

Variabel dan logika preprocessing yang Anda tentukan di luar fungsi handler tidak dipertahankan saat container baru dibuat.

- Fungsi berumur panjang (atau disematkan) dimulai saat perangkat lunak AWS IoT Greengrass Core dimulai dan dijalankan dalam satu wadah. Kontainer yang sama memproses semua data yang Anda kirim ke fungsi.

  Beberapa pemanggilan diantrian hingga perangkat lunak AWS IoT Greengrass Core menjalankan pemanggilan sebelumnya.

  Variabel dan logika preprocessing yang Anda definisikan di luar fungsi handler dipertahankan untuk setiap pemanggilan handler.

  Gunakan fungsi Lambda yang berumur panjang saat Anda harus mulai melakukan pekerjaan tanpa input awal apa pun. Misalnya, fungsi yang berumur panjang dapat memuat dan mulai memproses model pembelajaran mesin agar siap ketika fungsi menerima data perangkat.

  > **Note**
  >
  > Fungsi berumur panjang memiliki batas waktu yang terkait dengan setiap pemanggilan penangan mereka. Jika Anda ingin memanggil kode yang berjalan tanpa batas waktu, Anda harus memulainya di luar handler. Pastikan tidak ada kode pemblokiran di luar handler yang mungkin mencegah fungsi dari inisialisasi.
  > Fungsi-fungsi ini berjalan kecuali perangkat lunak AWS IoT Greengrass Core berhenti, seperti selama penerapan atau reboot. Fungsi-fungsi ini tidak akan berjalan jika fungsi tersebut menemukan pengecualian yang tidak tertangkap, melebihi batas memorinya, atau memasuki status kesalahan, seperti batas waktu penangan.

Untuk informasi selengkapnya tentang penggunaan kembali kontainer, lihat Memahami Penggunaan Kembali Kontainer AWS Lambda di Blog AWS Komputasi.

Konfigurasikan kontainerisasi fungsi Lambda

Secara default, fungsi Lambda berjalan di dalam wadah. AWS IoT Greengrass Wadah Greengrass memberikan isolasi antara fungsi Anda dan host. Isolasi ini meningkatkan keamanan untuk host dan fungsi dalam wadah.

Kami menyarankan Anda menjalankan fungsi Lambda dalam wadah Greengrass, kecuali jika kasus penggunaan Anda mengharuskannya berjalan tanpa kontainerisasi. Dengan menjalankan fungsi Lambda Anda dalam wadah Greengrass, Anda memiliki kontrol lebih besar atas bagaimana Anda membatasi akses ke sumber daya.

Anda dapat menjalankan fungsi Lambda tanpa containerization dalam kasus berikut:

- Anda ingin berjalan AWS IoT Greengrass di perangkat yang tidak mendukung mode penampung. Contohnya adalah jika Anda ingin menggunakan distribusi Linux khusus, atau memiliki versi kernel sebelumnya yang kedaluwarsa.
- 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 penerapan, atau yang jalurnya dapat berubah setelah penerapan. Contoh sumber daya ini adalah perangkat yang dapat dicolokkan.
- Anda memiliki aplikasi sebelumnya yang ditulis sebagai proses, dan Anda mengalami masalah saat menjalankannya di wadah Greengrass.

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 wadah Greengrass tidak memiliki akses ke kode yang diterapkan dari fungsi Lambda lainnya, meskipun dijalankan dengan grup sistem yang sama. Dengan kata lain, fungsi Lambda Anda berjalan dengan peningkatan isolasi satu sama lain. • Karena perangkat lunak AWS IoT Greengras s Core menjalankan semua proses anak dalam wadah yang sama dengan fungsi

Kontainerisasi	Catatan
	Lambda, proses anak berhenti ketika fungsi Lambda berhenti.

Tanpa kontainer	• Fitur berikut tidak tersedia untuk fungsi Lambda non-kontainer:
	  • Batas memori fungsi Lambda.
	  • Sumber daya perangkat dan volume lokal. Anda harus mengakses sumber daya ini menggunakan jalur file mereka di perangkat inti alih-alih sebagai sumber daya fungsi Lambda.
	• 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.
	• Fungsi Lambda non-kontainer memiliki akses hanya-baca ke kode yang diterapkan dari fungsi Lambda lain yang berjalan dengan grup sistem yang sama.

Jika Anda mengubah kontainerisasi untuk fungsi Lambda saat Anda menerapkannya, fungsi tersebut mungkin tidak berfungsi seperti yang diharapkan. Jika fungsi Lambda menggunakan sumber daya lokal yang tidak lagi tersedia dengan setelan kontainerisasi baru, penerapan gagal.

- Saat Anda mengubah fungsi Lambda dari berjalan di wadah Greengrass menjadi berjalan tanpa kontainerisasi, batas memori fungsi akan dibuang. Anda harus mengakses sistem file secara langsung daripada menggunakan sumber daya lokal terlampir. Anda harus menghapus sumber daya yang terlampir sebelum menerapkan fungsi Lambda.
- 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 batas memori 16 MB default. Anda dapat mengonfigurasi pengaturan ini untuk setiap fungsi Lambda saat Anda menerapkannya.

Untuk mengubah setelan kontainerisasi komponen fungsi Lambda, tetapkan nilai parameter `containerMode` konfigurasi ke salah satu opsi berikut saat Anda menerapkan komponen.

- `NoContainer` – Komponen tersebut tidak berjalan di lingkungan waktu aktif terisolasi.
- `GreengrassContainer` – Komponen tersebut berjalan di lingkungan waktu aktif yang terisolasi di dalam kontainer AWS IoT Greengrass.

Untuk informasi selengkapnya tentang cara menerapkan dan mengonfigurasi komponen, lihat Deploy komponen AWS IoT Greengrass ke perangkat danPerbarui konfigurasi komponen.

Impor fungsi Lambda sebagai komponen (konsol)

Saat Anda menggunakan konsol AWS IoT Greengrass tersebut untuk membuat komponen fungsi Lambda, Anda mengimpor elemen fungsi AWS Lambda yang ada dan kemudian mengonfigurasinya untuk membuat komponen yang berjalan pada perangkat Greengrass Anda.

Sebelum memulai, tinjau persyaratan untuk menjalankan fungsi Lambda pada perangkat Greengrass.

Tugas

- Langkah 1: Pilih fungsi Lambda yang akan diimpor
- Langkah 2: Konfigurasikan parameter fungsi Lambda
- Langkah 3: (Opsional) Tentukan platform yang didukung untuk fungsi Lambda
- Langkah 4: (Opsional) Tentukan dependensi komponen untuk fungsi Lambda
- Langkah 5: (Opsional) Jalankan fungsi Lambda dalam kontainer
- Langkah 6: Buat komponen fungsi Lambda

Langkah 1: Pilih fungsi Lambda yang akan diimpor

1. Pada menu navigasi konsol AWS IoT Greengrass, pilih Komponen.
2. Pada halaman Komponen, pilih Buat komponen.
3. Pada halaman Buat komponen, di bawah Informasi komponen, pilih Impor fungsi Lambda.
4. Pada Fungsi Lambda, cari dan pilih fungsi Lambda yang ingin Anda impor.

   AWS IoT Greengrass membuat komponen dengan nama fungsi Lambda.

5. Pada Versi fungsi Lambda, pilih versi yang akan diimpor. Anda tidak dapat memilih alias Lambda seperti `$LATEST`.

   AWS IoT Greengrass membuat komponen dengan versi fungsi Lambda sebagai versi semantik yang valid. Misalnya, jika versi fungsi Anda adalah `3`, versi komponen akan menjadi `3.0.0`.

Langkah 2: Konfigurasikan parameter fungsi Lambda

Pada halaman Buat komponen, di bawah Konfigurasi fungsi Lambda, konfigurasikan parameter berikut yang akan digunakan untuk menjalankan fungsi Lambda.

1. (Opsional) Tambahkan sumber peristiwa di mana fungsi Lambda berlangganan untuk pesan kerja. Anda dapat menentukan sumber acara untuk berlangganan fungsi ini ke pesan penerbitan/berlangganan lokal dan AWS IoT Core pesan MQTT. Fungsi Lambda dipanggil saat menerima pesan dari sumber peristiwa.

   > **Note**
   >
   > Untuk berlangganan fungsi ini ke pesan dari fungsi atau komponen Lambda lainnya, gunakan komponen router langganan lama saat Anda menerapkan komponen fungsi Lambda ini. Saat Anda men-deploy komponen router langganan warisan, tentukan langganan yang digunakan oleh fungsi Lambda tersebut.

   Di bawah Sumber peristiwa, lakukan hal berikut untuk menambahkan sumber peristiwa:

   a. Untuk setiap sumber peristiwa yang Anda tambahkan, tentukan opsi berikut:

      - Topik — Topik untuk berlangganan pesan.
      - Jenis — Jenis sumber peristiwa. Pilih dari salah satu pilihan berikut:
        - Publikasi/berlangganan lokal - Berlangganan pesan penerbitan/berlangganan lokal.

          Jika Anda menggunakan Greengrass nucleus v2.6.0 atau yang lebih baru dan Lambda manager v2.2.5 atau yang lebih baru, Anda dapat menggunakan wildcard topik MQTT (dan) di Topik saat Anda menentukan jenis ini. **+ #**
        - AWS IoT CoreMQTT — Berlangganan pesan MQTT. AWS IoT Core

Anda dapat menggunakan wildcard topik MQTT (+dan#) di Topik saat Anda menentukan jenis ini.

b. Untuk menambahkan sumber peristiwa lain, pilih Tambahkan sumber peristiwa dan ulangi langkah sebelumnya. Untuk menghapus sumber peristiwa, pilih Hapus di samping sumber peristiwa yang ingin Anda hapus.

2. Untuk Batas waktu (detik), masukkan jumlah maksimum waktu dalam detik yang dapoat dijalankan oleh fungsi Lambda yang tidak disematkan sebelum waktunya habis. Default-nya adalah 3 detik.

3. Untuk Yang Disematkan, pilih apakah komponen fungsi Lambda akan disematkan. Default-nya adalah BETUL.

   - Fungsi Lambda yang disematkan (atau berumur panjang) dimulai saat AWS IoT Greengrass dimulai dan terus berjalan di kontainernya sendiri.
   - Fungsi Lambda yang tidak disematkan (atau sesuai permintaan) dimulai hanya ketika menerima item pekerjaan dan keluar setelah tidak berjalan selama waktu tidak berjalan maksimum yang ditentukan. Jika fungsi tersebut memiliki beberapa item pekerjaan, perangkat lunak AWS IoT Greengrass akan membuat beberapa instans dari fungsi tersebut.

4. (Opsional) Di bawah Parameter tambahan, atur parameter fungsi Lambda berikut.

   - Batas waktu status (detik) — Interval dalam detik di mana komponen fungsi Lambda mengirimkan pembaruan status ke komponen manajer Lambda. Parameter ini hanya berlaku untuk fungsi yang disematkan. Default-nya adalah 60 detik.
   - Ukuran antrean maksimum – Ukuran maksimum antrean pesan untuk komponen fungsi Lambda. Perangkat lunak inti AWS IoT Greengrass menyimpan pesan dalam antrean FIFO (pertama-masuk-pertama-keluar) sampai dapat menjalankan fungsi Lambda untuk mengonsumsi setiap pesan. Default-nya adalah 1.000 pesan.
   - Jumlah maksimum instans — Jumlah maksimum instans yang dapat dijalankan oleh fungsi Lambda yang tidak disematkan pada saat yang sama. Default-nya adalah 100 instans.
   - Jumlah maksimum waktu henti (detik) — Jumlah maksimum waktu dalam detik di mana fungsi Lambda yang tidak disematkan dapat tidak bekerja sebelum perangkat lunak inti AWS IoT Greengrass menghentikan prosesnya. Bawaannya adalah 60 detik.
   - Jenis pengodean — Jenis muatan yang didukung oleh fungsi Lambda. Pilih dari salah satu pilihan berikut:
     - JSON

- Biner

Default-nya adalah JSON.

5. (Opsional) Tentukan daftar argumen baris perintah yang akan diteruskan ke fungsi Lambda ketika berjalan.

   a. Di bawah Parameter tambahan, Argumen proses, pilih Tambahkan argumen.

   b. Untuk setiap argumen yang Anda tambahkan, masukkan argumen yang ingin Anda teruskan ke fungsi.

   c. Untuk menghapus argumen, pilih Hapus di samping argumen yang ingin Anda hapus.

6. (Opsional) Tentukan variabel lingkungan yang tersedia untuk fungsi Lambda ketika berjalan. Variabel lingkungan memungkinkan Anda untuk menyimpan dan memperbarui pengaturan konfigurasi tanpa perlu mengubah kode fungsi.

   a. Di bawah Parameter tambahan, Variabel lingkungan, pilih Tambahkan variabel lingkungan.

   b. Untuk setiap variabel lingkungan yang Anda tambahkan, tentukan opsi berikut:

      - Kunci — Nama variabel.
      - Nilai — Nilai default untuk variabel ini.

   c. Untuk menghapus variabel lingkungan, pilih Hapus di samping variabel lingkungan yang ingin Anda hapus.

Langkah 3: (Opsional) Tentukan platform yang didukung untuk fungsi Lambda

Semua perangkat inti memiliki atribut untuk sistem operasi dan arsitektur. Ketika Anda men-deploy komponen fungsi Lambda, perangkat lunak inti AWS IoT Greengrass akan membandingkan nilai platform yang Anda tentukan dengan atribut platform pada perangkat inti untuk menentukan apakah fungsi Lambda didukung pada perangkat tersebut.

> **Note**
>
> Anda dapat menentukan atribut platform kustom untuk perangkat inti ketika Anda men-deploy komponen inti Greengrass ke perangkat inti. Untuk informasi lebih lanjut, lihat bagian platform menimpa parameter dari komponen nukleus Greengrass.

Di bawah Konfigurasi fungsi Lambda, Parameter tambahan, Platform, lakukan hal berikut untuk menentukan platform yang mendukung fungsi Lambda ini.

1. Untuk setiap platform, tentukan opsi berikut:
   - Sistem operasi – Nama sistem operasi untuk platform tersebut. Saat ini, satu-satunya nilai yang didukung adalah `linux`.
   - Arsitektur — Arsitektur prosesor untuk platform. Nilai yang didukung adalah:
     - `amd64`
     - `arm`
     - `aarch64`
     - `x86`
2. Untuk menambahkan platform lain, pilih Tambahkan platform dan ulangi langkah sebelumnya. Untuk menghapus platform yang didukung, pilih Hapus di samping platform yang ingin Anda hapus.

Langkah 4: (Opsional) Tentukan dependensi komponen untuk fungsi Lambda

Dependensi komponen mengidentifikasi tambahan komponen yang disediakan oleh AWS atau komponen kustom yang digunakan oleh fungsi Anda. Ketika Anda men-deploy komponen fungsi Lambda, deployment tersebut mencakup dependensi ini yang akan dijalankan oleh fungsi Anda.

> **Important**
>
> Untuk mengimpor fungsi Lambda yang Anda buat untuk dijalankan di AWS IoT Greengrass V1, Anda harus menentukan dependensi komponen individual untuk fitur yang digunakan fungsi Anda, seperti rahasia, bayangan lokal, dan pengelola aliran. Tentukan komponen ini sebagai dependensi keras sehingga komponen fungsi Lambda Anda akan me-restart jika dependensi tersebut mengubah keadaan. Untuk informasi selengkapnya, lihat Impor fungsi Lambda V1.

Di bawah Konfigurasi fungsi Lambda, Parameter tambahan, Dependensi komponen, lengkapi langkah-langkah berikut untuk menentukan dependensi komponen untuk fungsi Lambda Anda.

1. Pilih Tambahkan dependensi.
2. Untuk setiap komponen dependensi yang Anda tambahkan, tentukan opsi berikut:

   - Nama komponen – Nama komponen tersebut. Misalnya, masukkan **`aws.greengrass.StreamManager`** untuk menyertakan komponen stream manager.
   - Persyaratan versi – Kendala versi semantik npm-style yang mengidentifikasi versi yang kompatibel dari dependensi komponen ini. Anda dapat menentukan versi tunggal atau serangkaian versi. Misalnya, masukkan **`^1.0.0`** untuk menetapkan bahwa fungsi Lambda ini tergantung pada versi apa pun dalam versi utama pertama dari komponen stream manager. Untuk informasi lebih lanjut tentang kendala versi semantik, lihat kalkulator semver npm.
   - Jenis – Jenis dependensi. Pilih dari salah satu opsi berikut:
     - Keras – Komponen fungsi Lambda akan memulai ulang jika dependensi mengubah keadaan. Ini adalah pilihan default.
     - Lunak – Komponen fungsi Lambda tidak memulai ulang jika dependensi mengubah keadaan.
3. Untuk menghapus dependensi komponen, pilih Hapus di sebelah dependensi komponen

Langkah 5: (Opsional) Jalankan fungsi Lambda dalam kontainer

Secara default, fungsi Lambda berjalan di lingkungan waktu aktif yang terisolasi di dalam perangkat lunak inti AWS IoT Greengrass. Anda juga dapat memilih untuk menjalankan fungsi Lambda sebagai proses tanpa isolasi apa pun (yaitu, dalam mode Tanpa kontainer).

Di bawah Konfigurasi proses Linux, untuk Mode isolasi, pilih dari opsi berikut untuk memilih kontainerisasi untuk fungsi Lambda Anda:

- Kontainer Greengrass - Fungsi Lambda berjalan di kontainer. Ini adalah pilihan default.
- Tanpa kontainer — Fungsi Lambda berjalan sebagai proses tanpa isolasi apa pun.

Jika Anda menjalankan fungsi Lambda dalam kontainer, selesaikan langkah-langkah berikut untuk mengonfigurasi konfigurasi proses untuk fungsi Lambda.

1. Konfigurasikan jumlah memori dan sumber daya sistem, seperti volume dan perangkat, agar dapat tersedia untuk kontainer.

   Di bawah Parameter kontainer, lakukan hal berikut.

a. Untuk Ukuran memori, masukkan ukuran memori yang ingin Anda alokasikan ke kontainer. Anda dapat menentukan ukuran memori dalam MB atau kB.

b. Untuk Folder sys hanya-baca, pilih apakah kontainer tersebut dapat membaca informasi dari folder `/sys` perangkat tersebut. Default-nya adalah SALAH.

2. (Opsional) Konfigurasikan volume lokal yang dapat diakses oleh fungsi Lambda yang terkontainerisasi. Bila Anda menentukan volume, perangkat lunak inti AWS IoT Greengrass akan memasang file sumber ke tujuan di dalam kontainer.

   a. Di bawah volume, pilih Tambahkan volume.

   b. Untuk setiap volume yang Anda tambahkan, tentukan opsi berikut:

      - Volume fisik – Path ke folder sumber pada perangkat inti.
      - Volume logis – Path ke folder tujuan dalam kontainer.
      - Izin — (Opsional) Izin untuk mengakses folder sumber dari kontainer. Pilih dari salah satu pilihan berikut:
        - Hanya-baca — Fungsi Lambda memiliki akses hanya baca ke folder sumber. Ini adalah pilihan default.
        - Baca-tulis — Fungsi Lambda memiliki akses baca/tulis ke folder sumber.
      - Tambahkan pemilik grup — (Opsional) Apakah akan menambahkan grup sistem yang menjalankan komponen fungsi Lambda sebagai pemilik folder sumber atau tidak. Default-nya adalah SALAH.

   c. Untuk menghapus volume, pilih Hapus di samping volume yang ingin Anda hapus.

3. (Opsional) Konfigurasikan perangkat sistem lokal yang dapat diakses oleh fungsi Lambda yang terkontainerisasi.

   a. Di bawah Perangkat, pilih Tambahkan perangkat.

   b. Untuk setiap perangkat yang Anda tambahkan, tentukan opsi berikut:

      - Jalur pasang – Jalur ke perangkat sistem pada perangkat inti.
      - Izin – (Opsional) Izin untuk mengakses perangkat sistem dari kontainer. Pilih dari salah satu pilihan berikut:
        - Hanya-baca — Fungsi Lambda memiliki akses hanya baca ke perangkat sistem. Ini adalah pilihan default.
        - Baca-tulis — Fungsi Lambda memiliki akses baca/tulis ke folder sumber.

- Tambahkan pemilik grup — (Opsional) Apakah akan menambahkan grup sistem yang menjalankan komponen fungsi Lambda sebagai pemilik perangkat sistem atau tidak. Default-nya adalah SALAH.

Langkah 6: Buat komponen fungsi Lambda

Setelah Anda mengonfigurasi pengaturan untuk komponen fungsi Lambda Anda, pilih Buat untuk menyelesaikan pembuatan komponen baru.

Untuk menjalankan fungsi Lambda pada perangkat inti Anda, Anda dapat kemudian men-deploy komponen baru ke perangkat inti Anda. Lihat informasi yang lebih lengkap di Deploy komponen AWS IoT Greengrass ke perangkat.

Impor fungsi Lambda sebagai komponen (AWS CLI)

Gunakan CreateComponentVersionoperasi untuk membuat komponen dari fungsi Lambda. Saat Anda memanggil operasi ini, tentukan `lambdaFunction` untuk mengimpor fungsi Lambda.

Tugas

- Langkah 1: Tentukan konfigurasi fungsi Lambda
- Langkah 2: Buat komponen fungsi Lambda

Langkah 1: Tentukan konfigurasi fungsi Lambda

1. Buat file bernama `lambda-function-component.json`, lalu salin objek JSON berikut ke dalam file. Ganti `lambdaArn` dengan ARN fungsi Lambda yang akan diimpor.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1"
  }
}
```

> **Important**
>
> Anda harus menentukan ARN yang mencakup versi fungsi yang akan diimpor. Anda tidak dapat menggunakan alias versi seperti `$LATEST`.

2. (Opsional) Tentukan nama (`componentName`) komponen. Jika Anda menghilangkan parameter ini, AWS IoT Greengrass akan membuat komponen dengan nama fungsi Lambda.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda"
  }
}
```

3. (Opsional) Tentukan versi (`componentVersion`) untuk komponen. Jika Anda menghilangkan parameter ini, AWS IoT Greengrass akan membuat komponen dengan versi fungsi Lambda sebagai versi semantik yang valid. Misalnya, jika versi fungsi Anda adalah `3`, versi komponen akan menjadi `3.0.0`.

> **Note**
>
> Setiap versi komponen yang Anda upload harus unik. Pastikan Anda mengunggah versi komponen yang benar, karena Anda tidak dapat mengeditnya setelah mengunggahnya. AWS IoT Greengrass menggunakan versi semantik untuk komponen. Versi semantik mengikuti sistem nomor mayor.minor.patch. Sebagai contoh, versi `1.0.0` merupakan rilis mayor pertama untuk sebuah komponen. Untuk informasi lebih lanjut, lihat spesifikasi versi semantik.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0"
  }
}
```

4. (Opsional) Tentukan platform yang mendukung fungsi Lambda ini. Setiap platform berisi peta atribut yang menentukan suatu platform. Semua perangkat inti memiliki atribut untuk sistem operasi (`os`) dan arsitektur (`architecture`). Perangkat lunak inti AWS IoT Greengrass dapat menambahkan atribut platform lainnya. Anda dapat menentukan atribut platform kustom untuk perangkat inti ketika Anda men-deploy komponen inti Greengrass ke perangkat inti. Lakukan hal-hal berikut:

   a. Tambahkan daftar platform (`componentPlatforms`) pada fungsi Lambda di `lambda-function-component.json`.

   ```
   {
     "lambdaFunction": {
       "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
       "componentName": "com.example.HelloWorldLambda",
       "componentVersion": "1.0.0",
       "componentPlatforms": [

       ]
     }
   }
   ```

   b. Tambahkan setiap platform yang didukung ke daftar. Setiap platform memiliki `name` yang ramah untuk mengidentifikasinya dan peta atribut. Contoh berikut menetapkan bahwa fungsi ini mendukung perangkat x86 yang menjalankan Linux.

   ```
   {
     "name": "Linux x86",
     "attributes": {
       "os": "linux",
       "architecture": "x86"
     }
   }
   ```

   `lambda-function-component.json` Anda mungkin berisi dokumen yang mirip dengan contoh berikut.

   ```
   {
     "lambdaFunction": {
       "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
       "componentName": "com.example.HelloWorldLambda",
       "componentVersion": "1.0.0",
   ```

```
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ]
  }
}
```

5. (Opsional) Tentukan dependensi komponen untuk fungsi Lambda Anda. Ketika Anda men-deploy komponen fungsi Lambda, deployment tersebut mencakup dependensi ini yang akan dijalankan oleh fungsi Anda.

> **Important**
>
> Untuk mengimpor fungsi Lambda yang Anda buat untuk dijalankan di AWS IoT Greengrass V1, Anda harus menentukan dependensi komponen individual untuk fitur yang digunakan fungsi Anda, seperti rahasia, bayangan lokal, dan pengelola aliran. Tentukan komponen ini sebagai dependensi keras sehingga komponen fungsi Lambda Anda akan me-restart jika dependensi tersebut mengubah keadaan. Untuk informasi selengkapnya, lihat Impor fungsi Lambda V1.

Lakukan hal-hal berikut:

a. Tambahkan peta dependensi komponen (`componentDependencies`) pada fungsi Lambda di `lambda-function-component.json`.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
```

```
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {

    }
  }
}
```

b. Tambahkan dependensi pada setiap komponen pada peta. Tentukan nama komponen sebagai kunci dan tentukan objek dengan parameter berikut:

- `versionRequirement` – Kendala versi semantik npm-style yang menentukan versi komponen dependensi yang kompatibel. Anda dapat menentukan versi tunggal atau serangkaian versi. Untuk informasi lebih lanjut tentang kendala versi semantik, lihat kalkulator semver npm.
- `dependencyType` – (Opsional) Jenis dependensi. Pilih dari yang berikut ini:
  - `SOFT` – Komponen fungsi Lambda tidak memulai ulang jika dependensi mengubah keadaan.
  - `HARD` – Komponen fungsi Lambda memulai ulang jika dependensi mengubah keadaan.

  Default-nya adalah `HARD`.

Contoh berikut menetapkan bahwa fungsi Lambda ini tergantung pada versi apa pun dalam versi utama pertama dari komponen stream manager. Komponen fungsi Lambda akan me-restart ketika strean manager melakukan restart atau pembaruan.

```
{
  "aws.greengrass.StreamManager": {
    "versionRequirement": "^1.0.0",
    "dependencyType": "HARD"
  }
}
```

`lambda-function-component.json` Anda mungkin berisi dokumen yang mirip dengan contoh berikut.

```
{
```

```
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    }
  }
}
```

6. (Opsional) Konfigurasikan parameter fungsi Lambda yang akan digunakan untuk menjalankan fungsi. Anda dapat mengonfigurasi pilihan variabel lingkungan, sumber peristiwa pesan, batas waktu, dan pengaturan kontainer seperti ini. Lakukan hal-hal berikut:

   a. Tambahkan objek parameter Lambda (`componentLambdaParameters`) pada fungsi Lambda di `lambda-function-component.json`.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
```

```
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {

    }
  }
}
```

b. (Opsional) Tentukan sumber peristiwa di mana fungsi Lambda berlangganan untuk pesan kerja. Anda dapat menentukan sumber acara untuk berlangganan fungsi ini ke pesan penerbitan/berlangganan lokal dan AWS IoT Core pesan MQTT. Fungsi Lambda dipanggil saat menerima pesan dari sumber peristiwa.

> **Note**
>
> Untuk berlangganan fungsi ini ke pesan dari fungsi atau komponen Lambda lainnya, gunakan komponen router langganan lama saat Anda menerapkan komponen fungsi Lambda ini. Saat Anda men-deploy komponen router langganan warisan, tentukan langganan yang digunakan oleh fungsi Lambda tersebut.

Lakukan hal-hal berikut:

i. Tambahkan daftar sumber peristiwa (`eventSources`) pada parameter fungsi Lambda.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
```

```
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [

      ]
    }
  }
}
```

ii. Tambahkan setiap sumber peristiwa pada daftar. Setiap sumber peristiwa memiliki parameter berikut:

- `topic` — Topik untuk berlangganan pesan.
- `type` — Jenis sumber peristiwa. Pilih dari salah satu pilihan berikut:
  - `PUB_SUB` – Berlangganan ke pesan terbit/berlangganan lokal.

    Jika Anda menggunakan Greengrass nucleus v2.6.0 atau yang lebih baru dan Lambda manager v2.2.5 atau yang lebih baru, Anda dapat menggunakan wildcard topik MQTT (dan) di saat Anda menentukan jenis ini. `+` `#` `topic`
  - `IOT_CORE` – Berlangganan ke AWS IoT Core Pesan MQTT.

    Anda dapat menggunakan wildcard topik MQTT (+dan#) di `topic` saat Anda menentukan jenis ini.

Contoh berikut berlangganan MQTT AWS IoT Core pada topik yang sesuai dengan filter topik `hello/world/+`.

```
{
  "topic": "hello/world/+",
  "type": "IOT_CORE"
}
```

`lambda-function-component.json` Anda mungkin terlihat serupa dengan yang berikut ini.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ]
    }
  }
}
```

c. (Opsional) Tentukan salah satu parameter berikut di objek parameter fungsi Lambda:

- `environmentVariables` — Peta variabel lingkungan yang tersedia untuk fungsi Lambda ketika berjalan.
- `execArgs` — Daftar argumen yang akan diteruskan ke fungsi Lambda ketika berjalan.
- `inputPayloadEncodingType` — Jenis muatan yang didukung oleh fungsi Lambda. Pilih dari salah satu pilihan berikut:
  - `json`

  - `binary`

  Default: `json`

- `pinned` — Apakah fungsi Lambda akan disematkan atau tidak. Default-nya adalah `true`.
  - Fungsi Lambda yang disematkan (atau berumur panjang) dimulai saat AWS IoT Greengrass dimulai dan terus berjalan di kontainernya sendiri.
  - Fungsi Lambda yang tidak disematkan (atau sesuai permintaan) dimulai hanya ketika menerima item pekerjaan dan keluar setelah tidak berjalan selama waktu tidak berjalan maksimum yang ditentukan. Jika fungsi tersebut memiliki beberapa item pekerjaan, perangkat lunak AWS IoT Greengrass akan membuat beberapa instans dari fungsi tersebut.

  Gunakan `maxIdleTimeInSeconds` untuk mengatur waktu diam maksimum untuk fungsi Anda.
- `timeoutInSeconds` — Jumlah maksimum waktu dalam detik yang dapat dijalankan fungsi Lambda sebelum waktu habis. Default-nya adalah 3 detik.
- `statusTimeoutInSeconds` — Interval dalam detik di mana komponen fungsi Lambda mengirimkan update status ke komponen manajer Lambda. Parameter ini hanya berlaku pada fungsi yang disematkan. Waktu default adalah 60 detik.
- `maxIdleTimeInSeconds` — Jumlah maksimum waktu dalam detik di mana fungsi Lambda yang tidak disematkan dapat tidak bekerna sebelum perangkat lunak inti AWS IoT Greengrass menghentikan prosesnya. Bawaannya adalah 60 detik.
- `maxInstancesCount` — Jumlah maksimum contoh yang dapat dijalankan oleh fungsi Lambda yang tidak disematkan pada saat yang sama. Default-nya adalah 100 instans.
- `maxQueueSize` — Ukuran maksimum antrean pesan untuk komponen fungsi Lambda. Perangkat lunak AWS IoT Greengrass Core menyimpan pesan dalam antrian FIFO (first-in-first-out) hingga dapat menjalankan fungsi Lambda untuk mengkonsumsi setiap pesan. Default-nya adalah 1.000 pesan.

`lambda-function-component.json` Anda mungkin berisi dokumen yang mirip dengan contoh berikut.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
```

```
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500
    }
  }
}
```

d. (Opsional) Konfigurasikan pengaturan kontainer untuk fungsi Lambda. Secara default, fungsi Lambda berjalan di lingkungan waktu aktif yang terisolasi di dalam perangkat lunak inti AWS IoT Greengrass. Anda juga dapat memilih untuk menjalankan fungsi Lambda sebagai

proses tanpa isolasi apa pun. Jika Anda menjalankan fungsi Lambda dalam suatu kontainer, Anda akan mengonfigurasi ukuran memori kontainer dan sumber daya sistem apa yang tersedia untuk fungsi Lambda tersebut. Lakukan hal-hal berikut:

i. Tambahkan objek parameter proses Linux (`linuxProcessParams`) untuk objek parameter Lambda di `lambda-function-component.json`.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
```

```
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {

      }
    }
  }
}
```

ii. (Opsional) Tentukan apakah fungsi Lambda akan berjalan dalam suatu kontainer atau tidak. Tambahkan parameter `isolationMode` pada objek parameter proses, dan pilih dari opsi berikut:

- `GreengrassContainer` - Fungsi Lambda berjalan dalam sebuah kontainer.
- `NoContainer` — Fungsi Lambda berjalan sebagai proses tanpa isolasi apa pun.

Default-nya adalah `GreengrassContainer`.

iii. (Opsional) Jika Anda menjalankan fungsi Lambda dalam kontainer, Anda dapat mengonfigurasi jumlah memori dan sumber daya sistem, seperti volume dan perangkat, agar dapat tersedia untuk kontainer. Lakukan hal-hal berikut:

A. Tambahkan objek parameter kontainer (`containerParams`) pada objek parameter proses Lambda di `lambda-function-component.json`.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
```

```
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {
        "containerParams": {

        }
      }
    }
  }
}
```

B. (Opsional) Tambahkan parameter `memorySizeInKB` untuk menentukan ukuran memori kontainer. Default-nya adalah 16.384 KB (16 MB).

C. (Opsional) Tambahkan parameter `mountROSysfs` untuk menentukan apakah kontainer tersebut dapat membaca informasi dari folder perangkat `/sys` atau tidak. Default-nya adalah `false`.

D. (Opsional) Konfigurasikan volume lokal yang dapat diakses oleh fungsi Lambda yang terkontainerisasi. Bila Anda menentukan volume, perangkat lunak inti AWS

IoT Greengrass akan memasang file sumber ke tujuan di dalam kontainer. Lakukan hal-hal berikut:

I. Tambahkan daftar volume (`volumes`) pada parameter kontainer.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
```

```
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {
        "containerParams": {
          "memorySizeInKB": 32768,
          "mountROSysfs": true,
          "volumes": [

          ]
        }
      }
    }
  }
}
```

II. Tambahkan setiap volume ke daftar. Setiap volume memiliki parameter berikut:

- `sourcePath` – Path ke folder sumber pada perangkat inti.
- `destinationPath` – Path ke folder tujuan dalam kontainer.
- `permission` – (Opsional) Izin untuk mengakses folder sumber dari kontainer. Pilih dari salah satu pilihan berikut:
  - `ro` – Fungsi Lambda memiliki akses hanya-baca ke folder sumber.
  - `rw` – Fungsi Lambda memiliki akses baca-tulis ke folder sumber.

  Default-nya adalah `ro`.
- `addGroupOwner` – (Opsional) Apakah akan menambahkan grup sistem yang akan menjalankan fungsi Lambda sebagai pemilik folder sumber atau tidak. Default-nya adalah `false`.

`lambda-function-component.json` Anda mungkin berisi dokumen yang mirip dengan contoh berikut.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
```

```
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {
        "containerParams": {
          "memorySizeInKB": 32768,
          "mountROSysfs": true,
          "volumes": [
            {
              "sourcePath": "/var/data/src",
              "destinationPath": "/var/data/dest",
              "permission": "rw",
```

```
              "addGroupOwner": true
            }
          ]
        }
      }
    }
  }
}
```

E. (Opsional) Konfigurasikan perangkat sistem lokal yang dapat diakses oleh fungsi Lambda yang terkontainerisasi. Lakukan hal-hal berikut:

I. Tambahkan daftar volume (`devices`) pada parameter kontainer.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
```

```
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {
        "containerParams": {
          "memorySizeInKB": 32768,
          "mountROSysfs": true,
          "volumes": [
            {
              "sourcePath": "/var/data/src",
              "destinationPath": "/var/data/dest",
              "permission": "rw",
              "addGroupOwner": true
            }
          ],
          "devices": [

          ]
        }
      }
    }
  }
}
```

II. Tambahkan setiap perangkat sistem ke daftar. Setiap perangkat sistem memiliki parameter berikut:

- `path` – Path ke perangkat sistem pada perangkat inti.
- `permission` – (Opsional) Izin untuk mengakses perangkat sistem dari kontainer. Pilih dari salah satu pilihan berikut:
  - `ro` – Fungsi Lambda memiliki akses hanya-baca ke perangkat sistem.
  - `rw` – Fungsi Lambda memiliki akses baca-tulis ke perangkat sistem.

Default-nya adalah `ro`.

- `addGroupOwner` – (Opsional) Apakah akan menambahkan grup sistem yang akan menjalankan fungsi Lambda sebagai pemilik perangkat sistem atau tidak. Default-nya adalah `false`.

`lambda-function-component.json` Anda mungkin berisi dokumen yang mirip dengan contoh berikut.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
```

```
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {
        "containerParams": {
          "memorySizeInKB": 32768,
          "mountROSysfs": true,
          "volumes": [
            {
              "sourcePath": "/var/data/src",
              "destinationPath": "/var/data/dest",
              "permission": "rw",
              "addGroupOwner": true
            }
          ],
          "devices": [
            {
              "path": "/dev/sda3",
              "permission": "rw",
              "addGroupOwner": true
            }
          ]
        }
      }
    }
  }
}
```

7. (Opsional) Tambahkan tag (`tags`) untuk komponen tersebut. Untuk informasi selengkapnya, lihat Beri tag pada sumber daya AWS IoT Greengrass Version 2 Anda.

Langkah 2: Buat komponen fungsi Lambda

1. Jalankan perintah berikut untuk membuat komponen fungsi Lambda dari `lambda-function-component.json`.

```
aws greengrassv2 create-component-version --cli-input-json file://lambda-function-
component.json
```

Respons tersebut serupa dengan contoh berikut ini jika permintaannya berhasil.

```
{
  "arn":
 "arn:aws:greengrass:region:123456789012:components:com.example.HelloWorldLambda:versions:1
  "componentName": "com.example.HelloWorldLambda",
  "componentVersion": "1.0.0",
  "creationTimestamp": "Mon Dec 15 20:56:34 UTC 2020",
  "status": {
    "componentState": "REQUESTED",
    "message": "NONE",
    "errors": {}
  }
}
```

Salin `arn` dari output untuk memeriksa keadaan komponen pada langkah berikutnya.

2. Ketika Anda membuat komponen, keadaannya adalah `REQUESTED`. Kemudian, AWS IoT Greengrass akan memvalidasi bahwa komponen tersebut dapat di-deploy. Anda dapat menjalankan perintah berikut untuk melakukan kueri atas status komponen dan memverifikasi bahwa komponen Anda dapat di-deploy. Ganti `arn` dengan ARN dari langkah sebelumnya.

```
aws greengrassv2 describe-component \
  --arn "arn:aws:greengrass:region:account-
id:components:com.example.HelloWorldLambda:versions:1.0.0"
```

Jika komponen tervalidasi, respons akan menunjukkan bahwa keadaan komponen adalah `DEPLOYABLE`.

```
{
  "arn": "arn:aws:greengrass:region:account-
id:components:com.example.HelloWorldLambda:versions:1.0.0",
  "componentName": "com.example.HelloWorldLambda",
  "componentVersion": "1.0.0",
  "creationTimestamp": "2020-12-15T20:56:34.376000-08:00",
  "publisher": "AWS Lambda",
  "status": {
```

```
    "componentState": "DEPLOYABLE",
    "message": "NONE",
    "errors": {}
  },
  "platforms": [
    {
      "name": "Linux x86",
      "attributes": {
        "architecture": "x86",
        "os": "linux"
      }
    }
  ]
}
```

Setelah komponen adalah `DEPLOYABLE`, Anda dapat men-deploy fungsi Lambda ke perangkat inti Anda. Lihat informasi yang lebih lengkap di Deploy komponen AWS IoT Greengrass ke perangkat.

Gunakan AWS IoT Device SDK untuk berkomunikasi dengan inti Greengrass, komponen lain, dan AWS IoT Core

Komponen yang berjalan pada perangkat inti Anda dapat menggunakan pustaka komunikasi interproses AWS IoT Greengrass Inti (IPC) di dalam AWS IoT Device SDK untuk berkomunikasi dengan AWS IoT Greengrass inti dan komponen Greengrass lainnya. Untuk mengembangkan dan menjalankan komponen khusus yang menggunakan IPC, Anda harus menggunakan AWS IoT Device SDK untuk terhubung ke layanan AWS IoT Greengrass Core IPC dan melakukan operasi IPC.

Antarmuka IPC mendukung dua jenis operasi:

- Permintaan/tanggapan

  Komponen mengirim permintaan ke layanan IPC dan menerima respons yang berisi hasil permintaan.

- Berlangganan

  Komponen mengirim permintaan berlangganan ke layanan IPC dan mengharapkan aliran pesan peristiwa sebagai tanggapan. Komponen menyediakan bagian yang menangani langganan yang menangani pesan peristiwa, kesalahan, dan penutupan aliran. AWS IoT Device SDK Termasuk antarmuka handler dengan respon yang benar dan jenis peristiwa untuk setiap operasi IPC. Untuk informasi selengkapnya, lihat Berlangganan pengaliran peristiwa IPC.

Topik

- Berinteraksilah dengan konfigurasi komponen
- Ambil nilai-nilai rahasia
- Berinteraksi dengan bayangan lokal
- Kelola penerapan dan komponen lokal
- Otentikasi dan otorisasi perangkat klien

Versi klien IPC

Dalam versi yang lebih baru dari Java dan Python SDK, AWS IoT Greengrass menyediakan versi perbaikan dari klien IPC, yang disebut IPC client V2. Klien IPC V2:

- Mengurangi jumlah kode yang perlu Anda tulis untuk menggunakan operasi IPC dan membantu menghindari kesalahan umum yang dapat terjadi dengan klien IPC V1.
- Memanggil callback handler langganan di thread terpisah, sehingga Anda sekarang dapat menjalankan kode pemblokiran, termasuk panggilan fungsi IPC tambahan, dalam callback handler langganan. Klien IPC V1 menggunakan thread yang sama untuk berkomunikasi dengan server IPC dan memanggil callback handler langganan.
- Memungkinkan Anda memanggil operasi berlangganan menggunakan ekspresi Lambda (Java) atau fungsi (Python). Klien IPC V1 mengharuskan Anda untuk menentukan kelas handler berlangganan.
- Menyediakan versi sinkron dan asinkron dari setiap operasi IPC. Klien IPC V1 hanya menyediakan versi asinkron dari setiap operasi.

Kami menyarankan Anda menggunakan klien IPC V2 untuk memanfaatkan peningkatan ini. Namun, banyak contoh dalam dokumentasi ini dan dalam beberapa konten online hanya menunjukkan bagaimana menggunakan klien IPC V1. Anda dapat menggunakan contoh dan tutorial berikut untuk melihat komponen sampel yang menggunakan klien IPC V2:

- PublishToTopiccontoh
- SubscribeToTopiccontoh
- Tutorial: Mengembangkan komponen Greengrass yang menunda pembaruan komponen
- Tutorial: Berinteraksi dengan perangkat IoT lokal melalui MQTT

Saat ini, AWS IoT Device SDK untuk C++ v2 hanya mendukung klien IPC V1.

SDK yang didukung untuk komunikasi antar proses

Pustaka AWS IoT Greengrass Core IPC disertakan dalam versi berikut AWS IoT Device SDK .

SDK	Versi minimum	Penggunaan	
AWS IoT Device SDK untuk Java v2	v1.6.0	Lihat Gunakan AWS IoT Device SDK untuk Java v2 (klien IPC V2)	
AWS IoT Device SDK untuk Python v2	v1.9.0	Lihat Gunakan AWS IoT Device SDK untuk Python v2 (klien IPC V2)	
AWS IoT Device SDK untuk C++ v2	v1.17.0	Lihat Gunakan AWS IoT Device SDK untuk C++ v2	
AWS IoT Device SDK untuk JavaScript v2	v1.12.0	Lihat Gunakan AWS IoT Device SDK untuk JavaScript v2 (klien IPC V1)	

Connect ke layanan AWS IoT Greengrass Core IPC

Untuk menggunakan komunikasi interprocess dalam komponen kustom Anda, Anda harus membuat koneksi ke soket server IPC yang dijalankan oleh perangkat lunak AWS IoT Greengrass Core. Selesaikan tugas-tugas berikut untuk mengunduh dan menggunakan AWS IoT Device SDK dalam bahasa pilihan Anda.

Gunakan AWS IoT Device SDK untuk Java v2 (klien IPC V2)

Untuk menggunakan AWS IoT Device SDK untuk Java v2 (klien IPC V2)

1. Unduh AWS IoT Device SDK untuk Java v2 (v1.6.0 atau yang lebih baru).
2. Lakukan salah satu dari berikut ini untuk menjalankan kode kustom Anda dalam komponen Anda:

- Bangun komponen Anda sebagai file JAR yang menyertakan AWS IoT Device SDK, dan jalankan file JAR ini dalam resep komponen Anda.
- Tentukan AWS IoT Device SDK JAR sebagai artefak komponen, dan tambahkan artefak itu ke classpath saat Anda menjalankan aplikasi dalam resep komponen Anda.

3. Gunakan kode berikut untuk membuat klien IPC.

```
try (GreengrassCoreIPCClientV2 ipcClient =
 GreengrassCoreIPCClientV2.builder().build()) {
    // Use client.
} catch (Exception e) {
    LOGGER.log(Level.SEVERE, "Exception occurred when using IPC.", e);
    System.exit(1);
}
```

Gunakan AWS IoT Device SDK untuk Python v2 (klien IPC V2)

Untuk menggunakan AWS IoT Device SDK untuk Python v2 (klien IPC V2)

1. Unduh AWS IoT Device SDK untuk Python (v1.9.0 atau yang lebih baru).
2. Tambahkan langkah instalasi SDK ke siklus hidup instalasi dalam resep komponen Anda.
3. Buat koneksi ke layanan AWS IoT Greengrass Core IPC. Gunakan kode berikut untuk membuat klien IPC.

```
from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2

try:
    ipc_client = GreengrassCoreIPCClientV2()
    # Use IPC client.
except Exception:
    print('Exception occurred when using IPC.', file=sys.stderr)
    traceback.print_exc()
    exit(1)
```

Gunakan AWS IoT Device SDK untuk C++ v2

Untuk membangun AWS IoT Device SDK v2 untuk C ++, perangkat harus memiliki alat berikut:

- C++ 11 atau yang lebih baru
- CMake 3.1 atau yang lebih baru
- Salah satu penyusun berikut:
  - GCC 4.8 atau yang lebih baru
  - Clang 3.9 atau yang lebih baru
  - MSVC 2015 atau yang lebih baru

Untuk menggunakan AWS IoT Device SDK untuk C++ v2

1. Unduh AWS IoT Device SDK untuk C++ v2 (v1.17.0 atau yang lebih baru).
2. Ikuti petunjuk penginstalan di README untuk membangun C++ v2 dari sumber. AWS IoT Device SDK
3. Dalam alat bangun C ++ Anda, tautkan pustaka IPC Greengrass, `AWS::GreengrassIpc-cpp`, yang telah Anda bangun pada langkah sebelumnya. Contoh `CMakeLists.txt` berikut menautkan pustaka IPC Greengrass untuk proyek yang Anda bangun dengan CMake.

   ```
   cmake_minimum_required(VERSION 3.1)
   project (greengrassv2_pubsub_subscriber)

   file(GLOB MAIN_SRC
           "*.h"
           "*.cpp"
           )
   add_executable(${PROJECT_NAME} ${MAIN_SRC})

   set_target_properties(${PROJECT_NAME} PROPERTIES
           LINKER_LANGUAGE CXX
           CXX_STANDARD 11)
   find_package(aws-crt-cpp PATHS ~/sdk-cpp-workspace/build)
   find_package(EventstreamRpc-cpp PATHS ~/sdk-cpp-workspace/build)
   find_package(GreengrassIpc-cpp PATHS ~/sdk-cpp-workspace/build)
   target_link_libraries(${PROJECT_NAME} AWS::GreengrassIpc-cpp)
   ```

4. Dalam kode komponen Anda, buat koneksi ke layanan AWS IoT Greengrass Core IPC untuk membuat klien IPC ()`Aws::Greengrass::GreengrassCoreIpcClient`. Anda harus menentukan pengelola siklus hidup koneksi IPC yang menangani koneksi IPC, pemutusan, dan peristiwa kesalahan. Contoh berikut membuat klien IPC dan pengelola siklus hidup koneksi IPC yang mencetak ketika klien IPC terhubung, terputus, dan menemukan kesalahan.

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        std::cout << "OnConnectCallback" << std::endl;
    }

    void OnDisconnectCallback(RpcError error) override {
        std::cout << "OnDisconnectCallback: " << error.StatusToString() <<
 std::endl;
        exit(-1);
    }

    bool OnErrorCallback(RpcError error) override {
        std::cout << "OnErrorCallback: " << error.StatusToString() << std::endl;
        return true;
    }
};

int main() {
    // Create the IPC client.
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    // Use the IPC client to create an operation request.

    // Activate the operation request.
```

```
    auto activate = operation.Activate(request, nullptr);
    activate.wait();

    // Wait for Greengrass Core to respond to the request.
    auto responseFuture = operation.GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from
 Greengrass Core." << std::endl;
        exit(-1);
    }

    // Check the result of the request.
    auto response = responseFuture.get();
    if (response) {
        std::cout << "Successfully published to topic: " << topic << std::endl;
    } else {
        // An error occurred.
        std::cout << "Failed to publish to topic: " << topic << std::endl;
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            std::cout << "Operation error: " << error->GetMessage().value() <<
 std::endl;
        } else {
            std::cout << "RPC error: " << response.GetRpcError() << std::endl;
        }
        exit(-1);
    }

    return 0;
}
```

5. Untuk menjalankan kode kustom Anda dalam komponen Anda, bangun kode Anda sebagai artefak biner, dan jalankan artefak biner dalam resep komponen Anda. Atur `Execute` izin artefak `OWNER` untuk mengaktifkan perangkat lunak AWS IoT Greengrass Core menjalankan artefak biner.

   Bagian `Manifests` resep komponen Anda ini mungkin terlihat serupa dengan yang berikut ini.

   JSON

   ```
   {
   ```

```
  ...
  "Manifests": [
    {
      "Lifecycle": {
        "run": "{artifacts:path}/greengrassv2_pubsub_subscriber"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.PubSubSubscriberCpp/1.0.0/greengrassv2_pubsub_subscriber",
          "Permission": {
            "Execute": "OWNER"
          }
        }
      ]
    }
  ]
}
```

YAML

```
...
Manifests:
  - Lifecycle:
      run: {artifacts:path}/greengrassv2_pubsub_subscriber
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.PubSubSubscriberCpp/1.0.0/greengrassv2_pubsub_subscriber
        Permission:
          Execute: OWNER
```

Gunakan AWS IoT Device SDK untuk JavaScript v2 (klien IPC V1)

Untuk membangun AWS IoT Device SDK for JavaScript v2 untuk digunakan dengan NodeJS, perangkat harus memiliki alat berikut:

- NodeJS 10.0 atau yang lebih baru
  - Jalankan `node -v` untuk memeriksa versi Node.
- CMake 3.1 atau yang lebih baru

Untuk menggunakan AWS IoT Device SDK for JavaScript v2 (klien IPC V1)

1. Unduh AWS IoT Device SDK untuk JavaScript v2 (v1.12.10 atau yang lebih baru).
2. Ikuti petunjuk penginstalan di README untuk membangun AWS IoT Device SDK for JavaScript v2 dari sumber.
3. Buat koneksi ke layanan AWS IoT Greengrass Core IPC. Selesaikan langkah-langkah berikut untuk membuat klien IPC dan membuat sambungan.
4. Gunakan kode berikut untuk membuat klien IPC.

```
import * as greengrascoreipc from 'aws-iot-device-sdk-v2';

let client = greengrascoreipc.createClient();
```

5. Gunakan kode berikut untuk membuat sambungan dari komponen Anda ke inti Greengrass.

```
await client.connect();
```

Otorisasi komponen untuk melakukan operasi IPC

Untuk mengizinkan komponen kustom Anda untuk menggunakan beberapa operasi IPC, Anda harus menentukan kebijakan otorisasi yang memungkinkan komponen untuk melakukan operasi pada sumber daya tertentu. Setiap kebijakan otorisasi menentukan daftar operasi dan daftar sumber daya yang diizinkan oleh kebijakan. Sebagai contoh, layanan IPC olah pesan mempublikasikan/berlangganan menentukan operasi publikasi dan langganan untuk sumber daya topik. Anda dapat menentukan wildcard * untuk mengizinkan akses ke semua operasi atau semua sumber daya.

Anda menentukan kebijakan otorisasi dengan parameter `accessControl` konfigurasi, yang dapat Anda atur dalam resep komponen atau saat Anda menerapkan komponen. Objek `accessControl` memetakan pengidentifikasi layanan IPC pada daftar kebijakan otorisasi. Anda dapat menentukan beberapa kebijakan otorisasi untuk setiap layanan IPC untuk mengontrol akses. Setiap kebijakan otorisasi memiliki ID kebijakan, yang harus unik di antara semua komponen.

> **Tip**
>
> Untuk membuat ID kebijakan yang unik, Anda dapat menggabungkan nama komponen, nama layanan IPC, dan counter. Sebagai contoh, sebuah komponen bernama

`com.example.HelloWorld` mungkin menentukan dua kebijakan otorisasi publikasi/berlangganan dengan ID berikut:

- `com.example.HelloWorld:pubsub:1`
- `com.example.HelloWorld:pubsub:2`

Kebijakan otorisasi menggunakan format berikut. Objek ini adalah parameter konfigurasi `accessControl`.

JSON

```
{
  "IPC service identifier": {
    "policyId": {
      "policyDescription": "description",
      "operations": [
        "operation1",
        "operation2"
      ],
      "resources": [
        "resource1",
        "resource2"
      ]
    }
  }
}
```

YAML

```
IPC service identifier:
  policyId:
    policyDescription: description
    operations:
      - operation1
      - operation2
    resources:
      - resource1
      - resource2
```

Wildcard dalam kebijakan otorisasi

Anda dapat menggunakan * wildcard dalam `resources` elemen kebijakan otorisasi IPC untuk mengizinkan akses ke beberapa sumber daya dalam satu kebijakan otorisasi.

- Di semua versi inti Greengrass, Anda dapat menentukan * satu karakter sebagai sumber daya untuk memungkinkan akses ke semua sumber daya.
- Di Greengrass nucleus v2.6.0 dan yang lebih baru, Anda dapat menentukan karakter dalam sumber daya agar sesuai * dengan kombinasi karakter apa pun. Misalnya, Anda dapat menentukan `factory/1/devices/Thermostat*/status` untuk mengizinkan akses ke topik status untuk semua perangkat termostat di pabrik, tempat nama setiap perangkat dimulai`Thermostat`.

Saat menentukan kebijakan otorisasi untuk layanan AWS IoT Core MQTT IPC, Anda juga dapat menggunakan wildcard MQTT (dan) untuk mencocokkan beberapa sumber daya. + # Untuk informasi selengkapnya, lihat wildcard MQTT dalam kebijakan otorisasi AWS IoT Core MQTT IPC.

Variabel resep dalam kebijakan otorisasi

Jika Anda menggunakan Greengrass nucleus v2.6.0 atau yang lebih baru, dan Anda menyetel opsi konfigurasi Greengrass nucleus ke, Anda dapat menggunakan variabel resep dalam kebijakan interpolateComponentConfigurationotorisasi. `true{iot:thingName}` Bila Anda memerlukan kebijakan otorisasi yang menyertakan nama perangkat inti, seperti untuk topik MQTT atau bayangan perangkat, Anda dapat menggunakan variabel resep ini untuk mengonfigurasi kebijakan otorisasi tunggal untuk sekelompok perangkat inti. Misalnya, Anda dapat mengizinkan akses komponen ke sumber daya berikut untuk operasi IPC bayangan.

```
$aws/things/{iot:thingName}/shadow/
```

Karakter khusus dalam kebijakan otorisasi

Untuk menentukan literal * atau ? karakter dalam kebijakan otorisasi, Anda harus menggunakan urutan escape. Urutan escape berikut menginstruksikan perangkat lunak AWS IoT Greengrass Core untuk menggunakan nilai literal alih-alih makna khusus karakter. Misalnya, * karakter adalah wildcard yang cocok dengan kombinasi karakter apa pun.

Karakter literal	Karakter pelarian	Catatan
*	${*}	
?	${?}	AWS IoT Greengrass saat ini tidak mendukung ? wildcard, yang cocok dengan karakter tunggal mana pun.
$	${$}	Gunakan urutan escape ini untuk mencocokkan sumber daya yang berisi${. Misalnya, untuk mencocokkan sumber daya bernama${resourceName} , Anda harus menentukan${${$}{resourceName} . Jika tidak, untuk mencocokkan sumber daya yang berisi$, Anda dapat menggunakan literal$, seperti untuk mengizinkan akses ke topik yang dimulai dengan$aws.

Contoh kebijakan otorisasi

Anda dapat mereferensikan contoh kebijakan otorisasi berikut untuk membantu Anda mengonfigurasi kebijakan otorisasi untuk komponen Anda.

Example Contoh resep komponen dengan kebijakan otorisasi

Contoh resep komponen berikut meliputi objek `accessControl` yang menentukan kebijakan otorisasi. Kebijakan ini mengotorisasi komponen `com.example.HelloWorld` untuk dipublikasikan ke topik `test/topic`.

JSON

```
{
```

```
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.HelloWorld",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that publishes messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.HelloWorld:pubsub:1": {
            "policyDescription": "Allows access to publish to test/topic.",
            "operations": [
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "test/topic"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "java -jar {artifacts:path}/HelloWorld.jar"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.HelloWorld
ComponentVersion: '1.0.0'
ComponentDescription: A component that publishes messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
```

```
        "com.example.HelloWorld:pubsub:1":
          policyDescription: Allows access to publish to test/topic.
          operations:
            - "aws.greengrass#PublishToTopic"
          resources:
            - "test/topic"
Manifests:
  - Lifecycle:
      run: |-
        java -jar {artifacts:path}/HelloWorld.jar
```

Example Contoh pembaruan konfigurasi komponen dengan kebijakan otorisasi

Contoh pemutakhiran konfigurasi berikut dalam penerapan menentukan untuk mengonfigurasi komponen dengan `accessControl` objek yang mendefinisikan kebijakan otorisasi. Kebijakan ini mengotorisasi komponen `com.example.HelloWorld` untuk dipublikasikan ke topik `test/topic`.

Console

Konfigurasi untuk digabungkan

```
{
  "accessControl": {
    "aws.greengrass.ipc.pubsub": {
      "com.example.HelloWorld:pubsub:1": {
        "policyDescription": "Allows access to publish to test/topic.",
        "operations": [
          "aws.greengrass#PublishToTopic"
        ],
        "resources": [
          "test/topic"
        ]
      }
    }
  }
}
```

AWS CLI

Perintah berikut membuat penyebaran ke perangkat inti.

```
aws greengrassv2 create-deployment --cli-input-json file://hello-world-
deployment.json
```

`hello-world-deployment.json`File berisi dokumen JSON berikut.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.HelloWorld": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"accessControl\":{\"aws.greengrass.ipc.pubsub\":
{\"com.example.HelloWorld:pubsub:1\":{\"policyDescription\":\"Allows access to
 publish to test/topic.\",\"operations\":[\"aws.greengrass#PublishToTopic\"],
\"resources\":[\"test/topic\"]}}}}"
      }
    }
  }
}
```

Greengrass CLI

Perintah Greengrass CLI berikut membuat penerapan lokal pada perangkat inti.

```
sudo greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.HelloWorld=1.0.0" \
  --update-config hello-world-configuration.json
```

`hello-world-configuration.json`File berisi dokumen JSON berikut.

```
{
  "com.example.HelloWorld": {
    "MERGE": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.HelloWorld:pubsub:1": {
            "policyDescription": "Allows access to publish to test/topic.",
            "operations": [
```

```
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "test/topic"
            ]
          }
        }
      }
    }
  }
}
```

Berlangganan pengaliran peristiwa IPC

Anda dapat menggunakan operasi IPC untuk berlangganan aliran peristiwa pada perangkat inti Greengrass. Untuk menggunakan operasi berlangganan, tentukan penanganan langganan dan buat permintaan ke layanan IPC. Kemudian, klien IPC menjalankan fungsi penanganan langganan setiap kali perangkat inti mengalirkan pesan peristiwa ke komponen Anda.

Anda dapat menutup suatu langganan untuk menghentikan pemrosesan pesan peristiwa. Untuk melakukannya, hubungi `closeStream()` (Java), `close()` (Python), atau `Close()` (C++) pada objek operasi langganan yang Anda gunakan untuk membuka langganan.

Layanan AWS IoT Greengrass Core IPC mendukung operasi berlangganan berikut:

- SubscribeToTopic
- SubscribeToIoTCore
- SubscribeToComponentUpdates
- SubscribeToConfigurationUpdate
- SubscribeToValidateConfigurationUpdates

Topik

- Tentukan penanganan langganan
- Contoh penanganan langganan

Tentukan penanganan langganan

Untuk menentukan handler langganan, tentukan fungsi callback yang menangani pesan peristiwa, kesalahan, dan penutupan aliran. Jika Anda menggunakan klien IPC V1, Anda harus mendefinisikan fungsi-fungsi ini di kelas. Jika Anda menggunakan klien IPC V2, yang tersedia di versi SDK Java dan Python yang lebih baru, Anda dapat menentukan fungsi-fungsi ini tanpa membuat kelas handler berlangganan.

Java

Jika Anda menggunakan klien IPC V1, Anda harus mengimplementasikan antarmuka generik`software.amazon.awssdk.eventstreamrpc.StreamResponseHandler<`*`StreamEventT`*
*`StreamEventType`*adalah jenis pesan acara untuk operasi berlangganan. Tentukan fungsi berikut untuk menangani pesan peristiwa, kesalahan, dan penutupan aliran.

Jika Anda menggunakan klien IPC V2, Anda dapat menentukan fungsi-fungsi ini di luar kelas handler langganan atau menggunakan ekspresi lambda.

`void onStreamEvent(`*`StreamEventType`* `event)`

Callback yang dipanggil oleh klien IPC ketika menerima pesan peristiwa, seperti pesan MQTT atau notifikasi pembaruan komponen.

`boolean onStreamError(Throwable error)`

Callback yang dipanggil oleh klien IPC ketika terjadi kesalahan aliran.

Kembali BETUL untuk menutup aliran langganan sebagai akibat dari kesalahan, atau kembali SALAH untuk membuat aliran tetap terbuka.

`void onStreamClosed()`

Callback yang dipanggil oleh klien IPC ketika aliran menutup.

Python

Jika Anda menggunakan klien IPC V1, Anda harus memperluas kelas penangan respons aliran yang sesuai dengan operasi langganan. AWS IoT Device SDK Termasuk kelas handler berlangganan untuk setiap operasi berlangganan. *`StreamEventType`*adalah jenis pesan acara untuk operasi berlangganan. Tentukan fungsi berikut untuk menangani pesan peristiwa, kesalahan, dan penutupan aliran.

Jika Anda menggunakan klien IPC V2, Anda dapat menentukan fungsi-fungsi ini di luar kelas handler langganan atau menggunakan ekspresi lambda.

```
def on_stream_event(self, event: StreamEventType) -> None
```

Callback yang dipanggil oleh klien IPC ketika menerima pesan peristiwa, seperti pesan MQTT atau notifikasi pembaruan komponen.

```
def on_stream_error(self, error: Exception) -> bool
```

Callback yang dipanggil oleh klien IPC ketika terjadi kesalahan aliran.

Kembali BETUL untuk menutup aliran langganan sebagai akibat dari kesalahan, atau kembali SALAH untuk membuat aliran tetap terbuka.

```
def on_stream_closed(self) -> None
```

Callback yang dipanggil oleh klien IPC ketika aliran menutup.

C++

Terapkan kelas yang diturunkan dari penanganan respons aliran yang sesuai dengan operasi berlangganan. AWS IoT Device SDK Termasuk kelas dasar handler berlangganan untuk setiap operasi berlangganan. *StreamEventType*adalah jenis pesan acara untuk operasi berlangganan. Tentukan fungsi berikut untuk menangani pesan peristiwa, kesalahan, dan penutupan aliran.

```
void OnStreamEvent(StreamEventType *event)
```

Callback yang dipanggil oleh klien IPC ketika menerima pesan peristiwa, seperti pesan MQTT atau notifikasi pembaruan komponen.

```
bool OnStreamError(OperationError *error)
```

Callback yang dipanggil oleh klien IPC ketika terjadi kesalahan aliran.

Kembali BETUL untuk menutup aliran langganan sebagai akibat dari kesalahan, atau kembali SALAH untuk membuat aliran tetap terbuka.

```
void OnStreamClosed()
```

Callback yang dipanggil oleh klien IPC ketika aliran menutup.

JavaScript

Terapkan kelas yang diturunkan dari penanganan respons aliran yang sesuai dengan operasi berlangganan. AWS IoT Device SDK Termasuk kelas dasar handler berlangganan untuk setiap operasi berlangganan. *StreamEventType*adalah jenis pesan acara untuk operasi berlangganan. Tentukan fungsi berikut untuk menangani pesan peristiwa, kesalahan, dan penutupan aliran.

`on(event: 'ended', listener: StreamingOperationEndedListener)`

Callback yang dipanggil oleh klien IPC ketika aliran menutup.

`on(event: 'streamError', listener: StreamingRpcErrorListener)`

Callback yang dipanggil oleh klien IPC ketika terjadi kesalahan aliran.

Kembali BETUL untuk menutup aliran langganan sebagai akibat dari kesalahan, atau kembali SALAH untuk membuat aliran tetap terbuka.

`on(event: 'message', listener: (message: InboundMessageType) => void)`

Callback yang dipanggil oleh klien IPC ketika menerima pesan peristiwa, seperti pesan MQTT atau notifikasi pembaruan komponen.

Contoh penanganan langganan

Contoh berikut menunjukkan cara menggunakan operasi SubscribeToTopic dan penanganan langganan pada pesan publikasi/langganan lokal.

Java (IPC client V2)

Example Contoh: Berlangganan pesan publikasi/langganan lokal

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.SubscribeToTopicResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.*;

import java.nio.charset.StandardCharsets;
import java.util.Optional;

public class SubscribeToTopicV2 {
```

```
   public static void main(String[] args) {
       String topic = args[0];
       try (GreengrassCoreIPCClientV2 ipcClient =
GreengrassCoreIPCClientV2.builder().build()) {
           SubscribeToTopicRequest request = new
SubscribeToTopicRequest().withTopic(topic);
           GreengrassCoreIPCClientV2.StreamingResponse<SubscribeToTopicResponse,
                   SubscribeToTopicResponseHandler> response =
                   ipcClient.subscribeToTopic(request,
SubscribeToTopicV2::onStreamEvent,
                           Optional.of(SubscribeToTopicV2::onStreamError),
                           Optional.of(SubscribeToTopicV2::onStreamClosed));
           SubscribeToTopicResponseHandler responseHandler =
response.getHandler();
           System.out.println("Successfully subscribed to topic: " + topic);

           // Keep the main thread alive, or the process will exit.
           try {
               while (true) {
                   Thread.sleep(10000);
               }
           } catch (InterruptedException e) {
               System.out.println("Subscribe interrupted.");
           }

           // To stop subscribing, close the stream.
           responseHandler.closeStream();
       } catch (Exception e) {
           if (e.getCause() instanceof UnauthorizedError) {
               System.err.println("Unauthorized error while publishing to topic: "
+ topic);
           } else {
               System.err.println("Exception occurred when using IPC.");
           }
           e.printStackTrace();
           System.exit(1);
       }
   }

   public static void onStreamEvent(SubscriptionResponseMessage
subscriptionResponseMessage) {
       try {
           BinaryMessage binaryMessage =
subscriptionResponseMessage.getBinaryMessage();
```

```
            String message = new String(binaryMessage.getMessage(),
 StandardCharsets.UTF_8);
            String topic = binaryMessage.getContext().getTopic();
            System.out.printf("Received new message on topic %s: %s%n", topic,
 message);
        } catch (Exception e) {
            System.err.println("Exception occurred while processing subscription
 response " +
                    "message.");
            e.printStackTrace();
        }
    }

    public static boolean onStreamError(Throwable error) {
        System.err.println("Received a stream error.");
        error.printStackTrace();
        return false; // Return true to close stream, false to keep stream open.
    }

    public static void onStreamClosed() {
        System.out.println("Subscribe to topic stream closed.");
    }
}
```

Python (IPC client V2)

Example Contoh: Berlangganan pesan publikasi/langganan lokal

```
import sys
import time
import traceback

from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
from awsiot.greengrasscoreipc.model import (
    SubscriptionResponseMessage,
    UnauthorizedError
)


def main():
    args = sys.argv[1:]
    topic = args[0]

    try:
```

```
        ipc_client = GreengrassCoreIPCClientV2()
        # Subscription operations return a tuple with the response and the
 operation.
        _, operation = ipc_client.subscribe_to_topic(topic=topic,
 on_stream_event=on_stream_event,

 on_stream_error=on_stream_error, on_stream_closed=on_stream_closed)
        print('Successfully subscribed to topic: ' + topic)

        # Keep the main thread alive, or the process will exit.
        try:
            while True:
                time.sleep(10)
        except InterruptedError:
            print('Subscribe interrupted.')

        # To stop subscribing, close the stream.
        operation.close()
    except UnauthorizedError:
        print('Unauthorized error while subscribing to topic: ' +
              topic, file=sys.stderr)
        traceback.print_exc()
        exit(1)
    except Exception:
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)


def on_stream_event(event: SubscriptionResponseMessage) -> None:
    try:
        message = str(event.binary_message.message, 'utf-8')
        topic = event.binary_message.context.topic
        print('Received new message on topic %s: %s' % (topic, message))
    except:
        traceback.print_exc()


def on_stream_error(error: Exception) -> bool:
    print('Received a stream error.', file=sys.stderr)
    traceback.print_exc()
    return False  # Return True to close stream, False to keep stream open.
```

```
def on_stream_closed() -> None:
    print('Subscribe to topic stream closed.')


if __name__ == '__main__':
    main()
```

C++

Example Contoh: Berlangganan pesan publikasi/langganan lokal

```
#include <iostream>

#include </crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class SubscribeResponseHandler : public SubscribeToTopicStreamHandler {
    public:
        virtual ~SubscribeResponseHandler() {}

    private:
        void OnStreamEvent(SubscriptionResponseMessage *response) override {
            auto jsonMessage = response->GetJsonMessage();
            if (jsonMessage.has_value() &&
 jsonMessage.value().GetMessage().has_value()) {
                auto messageString =
 jsonMessage.value().GetMessage().value().View().WriteReadable();
                // Handle JSON message.
            } else {
                auto binaryMessage = response->GetBinaryMessage();
                if (binaryMessage.has_value() &&
 binaryMessage.value().GetMessage().has_value()) {
                    auto messageBytes = binaryMessage.value().GetMessage().value();
                    std::string messageString(messageBytes.begin(),
 messageBytes.end());
                    // Handle binary message.
                }
            }
        }

        bool OnStreamError(OperationError *error) override {
```

```
            // Handle error.
            return false; // Return true to close stream, false to keep stream open.
        }

        void OnStreamClosed() override {
            // Handle close.
        }
};

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
};

int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String topic("my/topic");
    int timeout = 10;

    SubscribeToTopicRequest request;
    request.SetTopic(topic);
```

```
    //SubscribeResponseHandler streamHandler;
    auto streamHandler = MakeShared<SubscribeResponseHandler>(DefaultAllocator());
    auto operation = ipcClient.NewSubscribeToTopic(streamHandler);
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
 Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
        } else {
            // Handle RPC error.
        }
        exit(-1);
    }

    // Keep the main thread alive, or the process will exit.
    while (true) {
        std::this_thread::sleep_for(std::chrono::seconds(10));
    }

    operation->Close();
    return 0;
}
```

JavaScript

Example Contoh: Berlangganan pesan publikasi/langganan lokal

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
```

```
import {SubscribeToTopicRequest, SubscriptionResponseMessage} from "aws-iot-device-
sdk-v2/dist/greengrasscoreipc/model";
import {RpcError} from "aws-iot-device-sdk-v2/dist/eventstream_rpc";

class SubscribeToTopic {
    private ipcClient : greengrasscoreipc.Client
    private readonly topic : string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.subscribeToTopic().then(r => console.log("Started workflow"));
    }

    private async subscribeToTopic() {
        try {
            this.ipcClient = await getIpcClient();

            const subscribeToTopicRequest : SubscribeToTopicRequest = {
                topic: this.topic,
            }

            const streamingOperation =
 this.ipcClient.subscribeToTopic(subscribeToTopicRequest, undefined); //
 conditionally apply options

            streamingOperation.on("message", (message: SubscriptionResponseMessage)
 => {
                // parse the message depending on your use cases, e.g.
                if(message.binaryMessage && message.binaryMessage.message) {
                    const receivedMessage =
 message.binaryMessage?.message.toString();
                }
            });

            streamingOperation.on("streamError", (error : RpcError) => {
                // define your own error handling logic
            })

            streamingOperation.on("ended", () => {
                // define your own logic
            })

            await streamingOperation.activate();
```

```
            // Keep the main thread alive, or the process will exit.
            await new Promise((resolve) => setTimeout(resolve, 10000))
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}

export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

// starting point
const subscribeToTopic = new SubscribeToTopic();
```

Praktik terbaik IPC

Praktik terbaik untuk menggunakan IPC dalam komponen khusus berbeda antara klien IPC V1 dan klien IPC V2. Ikuti praktik terbaik untuk versi klien IPC yang Anda gunakan.

IPC client V2

Klien IPC V2 menjalankan fungsi callback di thread terpisah, jadi dibandingkan dengan klien IPC V1, ada lebih sedikit pedoman yang harus Anda ikuti saat Anda menggunakan IPC dan menulis fungsi penangan langganan.

- Gunakan kembali satu klien IPC

Setelah Anda membuat klien IPC, tetap buka dan gunakan kembali untuk semua operasi IPC. Membuat beberapa klien menggunakan sumber daya tambahan dan dapat mengakibatkan kebocoran sumber daya.

- Tangani pengecualian

  Klien IPC V2 mencatat pengecualian yang tidak tertangkap dalam fungsi penangan langganan. Anda harus menangkap pengecualian dalam fungsi handler Anda untuk menangani kesalahan yang terjadi dalam kode Anda.

IPC client V1

Klien IPC V1 menggunakan utas tunggal yang berkomunikasi dengan server IPC dan memanggil penangan langganan. Anda harus mempertimbangkan perilaku sinkron ini ketika Anda menuliskan fungsi penanganan langganan.

- Gunakan kembali satu klien IPC

  Setelah Anda membuat klien IPC, tetap buka dan gunakan kembali untuk semua operasi IPC. Membuat beberapa klien menggunakan sumber daya tambahan dan dapat mengakibatkan kebocoran sumber daya.

- Jalankan kode pemblokiran secara asinkron

  Klien IPC V1 tidak dapat mengirim permintaan baru atau memproses pesan acara baru saat utas diblokir. Anda harus menjalankan kode pemblokiran di thread terpisah yang Anda jalankan dari fungsi handler. Kode pemblokiran meliputi panggilan `sleep`, loop yang terus berjalan, dan permintaan I/O sinkron yang membutuhkan waktu untuk diselesaikan.

- Kirim permintaan IPC baru secara asinkron

  Klien IPC V1 tidak dapat mengirim permintaan baru dari dalam fungsi handler langganan, karena permintaan memblokir fungsi handler jika Anda menunggu respons. Anda harus mengirim permintaan IPC di thread terpisah yang Anda jalankan dari fungsi handler.

- Tangani pengecualian

  Klien IPC V1 tidak menangani pengecualian yang tidak tertangkap dalam fungsi penangan langganan. Jika fungsi penanganan Anda melempar pengecualian, langganan tersebut akan menutup, dan pengecualian tidak akan muncul dalam log komponen Anda. Anda harus

menangkap pengecualian dalam fungsi handler Anda untuk menjaga langganan tetap terbuka dan mencatat kesalahan yang terjadi dalam kode Anda.

Pesan lokal publikasi/berlangganan

Olah pesan publikasi/berlangganan (pubsub) memungkinkan Anda untuk mengirim dan menerima pesan ke topik. Komponen dapat mempublikasikan pesan pada topik untuk mengirim pesan ke komponen lain. Kemudian, komponen yang berlangganan topik itu dapat bertindak atas pesan yang diterimanya.

> **Note**
>
> Anda tidak dapat menggunakan layanan IPC publikasi/berlangganan ini untuk mempublikasikan atau berlangganan MQTT AWS IoT Core . Untuk informasi selengkapnya tentang cara bertukar pesan dengan AWS IoT Core MQTT, lihat. Terbitkan/berlangganan pesan MQTT AWS IoT Core

Topik

SDK (Versi Minimum)

Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk mempublikasikan dan berlangganan pesan ke dan dari topik lokal.

SDK	Versi minimum	
AWS IoT Device SDK untuk Java v2	v1.2.10	

SDK	Versi minimum	
AWS IoT Device SDK untuk Python v2	v1.5.3	
AWS IoT Device SDK untuk C ++ v2	v1.17.0	
AWS IoT Device SDK for JavaScript v2	v1.12.0	

Otorisasi

Untuk menggunakan pesan publish/subscribe lokal dalam komponen kustom, Anda harus menentukan kebijakan otorisasi yang memungkinkan komponen Anda mengirim dan menerima pesan ke topik. Untuk informasi tentang cara menentukan kebijakan otorisasi, lihat Otorisasi komponen untuk melakukan operasi IPC.

Kebijakan otorisasi untuk olah pesan MQTT memiliki properti berikut.

Pengenal Layanan: `aws.greengrass.ipc.pubsub`

Operasi	Deskripsi	Sumber daya
`aws.greengrass#PublishToTopic`	Memungkinkan komponen untuk mempublikasikan pesan ke topik MQTT yang Anda tentukan.	String topik, seperti`test/topic` . Gunakan * untuk mencocokkan kombinasi karakter apa pun dalam suatu topik. String topik ini mendukung wildcard topik MQTT (# dan +).
`aws.greengrass#SubscribeToTopic`	Memungkinkan komponen untuk berlangganan pesan untuk topik yang Anda tentukan.	String topik, seperti`test/topic` . Gunakan * untuk mencocokkan kombinasi

Operasi	Deskripsi	Sumber daya
		karakter apa pun dalam suatu topik. Di Greengrass nucleus v2.6.0 dan yang lebih baru, Anda dapat berlangganan topik yang berisi wildcard topik MQTT (dan). # + String topik ini mendukung wildcard topik MQTT sebagai karakter literal. Misalnya, jika kebijakan otorisasi komponen memberikan akses ke`test/topic/#` , komponen dapat berlangganan`test/topic/#`, tetapi tidak dapat berlangganan. `test/topic/filter`

Operasi	Deskripsi	Sumber daya
*	Memungkinkan komponen untuk mempublikasikan dan berlanggan pesan untuk topik yang Anda tentukan.	String topik, sepertitest/topic . Gunakan * untuk mencocokkan kombinasi karakter apa pun dalam suatu topik. Di Greengrass nucleus v2.6.0 dan yang lebih baru, Anda dapat berlangganan topik yang berisi wildcard topik MQTT (dan). # + String topik ini mendukung wildcard topik MQTT sebagai karakter literal. Misalnya, jika kebijakan otorisasi komponen memberikan akses ketest/topic/# , komponen dapat berlangganantest/topic/#, tetapi tidak dapat berlangganan. test/topic/filter

Contoh kebijakan otorisasi

Anda dapat mereferensikan contoh kebijakan otorisasi berikut untuk membantu Anda mengonfigurasi kebijakan otorisasi untuk komponen Anda.

Example Contoh kebijakan otorisasi

Kebijakan otorisasi contoh berikut ini memungkinkan komponen untuk mempublikasikan dan berlangganan semua topik.

```
{
  "accessControl": {
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyLocalPubSubComponent:pubsub:1": {
        "policyDescription": "Allows access to publish/subscribe to all topics.",
```

```
        "operations": [
          "aws.greengrass#PublishToTopic",
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
}
```

PublishToTopic

Publikasikan pesan ke topik.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`topic`

Topik yang pesannya dipublikasikan.

`publishMessage`(Python:) `publish_message`

Pesan yang akan dipublikasikan. Objek ini, `PublishMessage`, berisi informasi berikut. Anda harus menentukan salah satu dari `jsonMessage` dan `binaryMessage`.

`jsonMessage`(Python:) `json_message`

(Opsional) Sebuah pesan JSON. Objek ini, `JsonMessage`, berisi informasi berikut:

`message`

Pesan JSON sebagai objek.

`context`

Konteks pesan, seperti topik di mana pesan itu diterbitkan.

Fitur ini tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass. Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk mengakses konteks pesan.

SDK	Versi minimum	
AWS IoT Device SDK untuk Java v2	v1.9.3	
AWS IoT Device SDK untuk Python v2	v1.11.3	
AWS IoT Device SDK untuk C++ v2	v1.18.4	
AWS IoT Device SDK for JavaScript v2	v1.12.0	

> **Note**
>
> Perangkat lunak AWS IoT Greengrass Core menggunakan objek pesan yang sama dalam `PublishToTopic` dan `SubscribeToTopic` operasi. Perangkat lunak AWS IoT Greengrass Core menetapkan objek konteks ini dalam pesan saat Anda berlangganan, dan mengabaikan objek konteks ini dalam pesan yang Anda terbitkan.

Objek ini, `MessageContext`, berisi informasi berikut:

`topic`

Topik di mana pesan itu diterbitkan.

`binaryMessage`(Python:) `binary_message`

(Opsional) Sebuah pesan biner. Objek ini, `BinaryMessage`, berisi informasi berikut:

`message`

Pesan biner sebagai gumpalan.

`context`

Konteks pesan, seperti topik di mana pesan itu diterbitkan.

Fitur ini tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass. Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk mengakses konteks pesan.

SDK	Versi minimum
AWS IoT Device SDK untuk Java v2	v1.9.3
AWS IoT Device SDK untuk Python v2	v1.11.3
AWS IoT Device SDK untuk C++ v2	v1.18.4
AWS IoT Device SDK for JavaScript v2	v1.12.0

> **Note**
>
> Perangkat lunak AWS IoT Greengrass Core menggunakan objek pesan yang sama dalam `PublishToTopic` dan `SubscribeToTopic` operasi. Perangkat lunak AWS IoT Greengrass Core menetapkan objek konteks ini dalam pesan saat Anda berlangganan, dan mengabaikan objek konteks ini dalam pesan yang Anda terbitkan.

Objek ini, `MessageContext`, berisi informasi berikut:

`topic`

Topik di mana pesan itu diterbitkan.

Respons

Operasi ini tidak memberikan informasi apa pun dalam tanggapannya.

Contoh

Contoh-contoh berikut ini menunjukkan cara memanggil operasi ini dalam kode komponen kustom.

Java (IPC client V2)

Example Contoh: Publikasikan pesan biner

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.model.BinaryMessage;
import software.amazon.awssdk.aws.greengrass.model.PublishMessage;
import software.amazon.awssdk.aws.greengrass.model.PublishToTopicRequest;
import software.amazon.awssdk.aws.greengrass.model.PublishToTopicResponse;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;

import java.nio.charset.StandardCharsets;

public class PublishToTopicV2 {

    public static void main(String[] args) {
        String topic = args[0];
        String message = args[1];
        try (GreengrassCoreIPCClientV2 ipcClient =
 GreengrassCoreIPCClientV2.builder().build()) {
            PublishToTopicV2.publishBinaryMessageToTopic(ipcClient, topic,
 message);
            System.out.println("Successfully published to topic: " + topic);
        } catch (Exception e) {
            if (e.getCause() instanceof UnauthorizedError) {
                System.err.println("Unauthorized error while publishing to topic: "
 + topic);
            } else {
                System.err.println("Exception occurred when using IPC.");
            }
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static PublishToTopicResponse publishBinaryMessageToTopic(
            GreengrassCoreIPCClientV2 ipcClient, String topic, String message)
 throws InterruptedException {
```

```
        BinaryMessage binaryMessage =
                new
 BinaryMessage().withMessage(message.getBytes(StandardCharsets.UTF_8));
        PublishMessage publishMessage = new
 PublishMessage().withBinaryMessage(binaryMessage);
        PublishToTopicRequest publishToTopicRequest =
                new
 PublishToTopicRequest().withTopic(topic).withPublishMessage(publishMessage);
        return ipcClient.publishToTopic(publishToTopicRequest);
    }
}
```

Python (IPC client V2)

Example Contoh: Publikasikan pesan biner

```
import sys
import traceback

from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
from awsiot.greengrasscoreipc.model import (
    PublishMessage,
    BinaryMessage
)


def main():
    args = sys.argv[1:]
    topic = args[0]
    message = args[1]

    try:
        ipc_client = GreengrassCoreIPCClientV2()
        publish_binary_message_to_topic(ipc_client, topic, message)
        print('Successfully published to topic: ' + topic)
    except Exception:
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)


def publish_binary_message_to_topic(ipc_client, topic, message):
    binary_message = BinaryMessage(message=bytes(message, 'utf-8'))
    publish_message = PublishMessage(binary_message=binary_message)
```

```
    return ipc_client.publish_to_topic(topic=topic,
 publish_message=publish_message)


if __name__ == '__main__':
    main()
```

C++

Example Contoh: Publikasikan pesan biner

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
};

int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
```

```
        std::cerr << "Failed to establish IPC connection: " <<
connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String topic("my/topic");
    String message("Hello, World!");
    int timeout = 10;

    PublishToTopicRequest request;
    Vector<uint8_t> messageData({message.begin(), message.end()});
    BinaryMessage binaryMessage;
    binaryMessage.SetMessage(messageData);
    PublishMessage publishMessage;
    publishMessage.SetBinaryMessage(binaryMessage);
    request.SetTopic(topic);
    request.SetPublishMessage(publishMessage);

    auto operation = ipcClient.NewPublishToTopic();
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
        } else {
            // Handle RPC error.
        }
    }
    return 0;
```

```
}
```

JavaScript

Example Contoh: Publikasikan pesan biner

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
import {BinaryMessage, PublishMessage, PublishToTopicRequest} from "aws-iot-device-
sdk-v2/dist/greengrasscoreipc/model";

class PublishToTopic {
    private ipcClient : greengrasscoreipc.Client
    private readonly topic : string;
    private readonly messageString : string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.messageString = "<define_your_message_string>";
        this.publishToTopic().then(r => console.log("Started workflow"));
    }

    private async publishToTopic() {
        try {
            this.ipcClient = await getIpcClient();

            const binaryMessage : BinaryMessage = {
                message: this.messageString
            }

            const publishMessage : PublishMessage = {
                binaryMessage: binaryMessage
            }

            const request : PublishToTopicRequest = {
                topic: this.topic,
                publishMessage: publishMessage
            }

            this.ipcClient.publishToTopic(request).finally(() =>
 console.log(`Published message ${publishMessage.binaryMessage?.message} to topic`))
```

```
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}


export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

// starting point
const publishToTopic = new PublishToTopic();
```

SubscribeToTopic

Berlangganan pesan tentang suatu topik.

Operasi ini adalah operasi berlangganan di mana Anda berlangganan aliran pesan peristiwa. Untuk menggunakan operasi ini, tentukan bagian yang menangani respons aliran dengan fungsi yang menangani pesan peristiwa, kesalahan, dan penutupan aliran. Untuk informasi selengkapnya, lihat Berlangganan pengaliran peristiwa IPC.

Jenis pesan peristiwa: `SubscriptionResponseMessage`

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`topic`

Topik yang harus dijadikan langganan.

> **Note**
>
> Di Greengrass nucleus v2.6.0 dan yang lebih baru, topik ini mendukung wildcard topik MQTT (dan). # +

`receiveMode`(Python:) `receive_mode`

(Opsional) Perilaku yang menentukan apakah komponen menerima pesan dari dirinya sendiri. Anda dapat mengubah perilaku ini agar komponen dapat bertindak berdasarkan pesannya sendiri. Perilaku default tergantung pada apakah topik berisi wildcard MQTT. Pilih dari salah satu pilihan berikut:

- `RECEIVE_ALL_MESSAGES`— Menerima semua pesan yang cocok dengan topik, termasuk pesan dari komponen yang berlangganan.

  Mode ini adalah opsi default saat Anda berlangganan topik yang tidak berisi wildcard MQTT.
- `RECEIVE_MESSAGES_FROM_OTHERS`— Menerima semua pesan yang cocok dengan topik, kecuali pesan dari komponen yang berlangganan.

  Mode ini adalah opsi default saat Anda berlangganan topik yang berisi wildcard MQTT.

Fitur ini tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass. Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk mengatur mode terima.

SDK	Versi minimum	
AWS IoT Device SDK untuk Java v2	v1.9.3	
AWS IoT Device SDK untuk Python v2	v1.11.3	
AWS IoT Device SDK untuk C++ v2	v1.18.4	

SDK	Versi minimum	
AWS IoT Device SDK untuk JavaScript v2	v1.12.0	

Respons

Tanggapan operasi ini memiliki informasi berikut:

`messages`

Aliran pesan. Objek ini, `SubscriptionResponseMessage`, berisi informasi berikut. Setiap pesan berisi `jsonMessage` atau `binaryMessage`.

`jsonMessage`(Python:) `json_message`

(Opsional) Sebuah pesan JSON. Objek ini, `JsonMessage`, berisi informasi berikut:

`message`

Pesan JSON sebagai objek.

`context`

Konteks pesan, seperti topik di mana pesan itu diterbitkan.

Fitur ini tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass. Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk mengakses konteks pesan.

SDK	Versi minimum	
AWS IoT Device SDK untuk Java v2	v1.9.3	
AWS IoT Device SDK untuk Python v2	v1.11.3	
AWS IoT Device SDK untuk C++ v2	v1.18.4	

SDK	Versi minimum	
AWS IoT Device SDK for JavaScript v2	v1.12.0	

> **Note**
>
> Perangkat lunak AWS IoT Greengrass Core menggunakan objek pesan yang sama dalam `PublishToTopic` dan `SubscribeToTopic` operasi. Perangkat lunak AWS IoT Greengrass Core menetapkan objek konteks ini dalam pesan saat Anda berlangganan, dan mengabaikan objek konteks ini dalam pesan yang Anda terbitkan.

Objek ini, `MessageContext`, berisi informasi berikut:

`topic`

Topik di mana pesan itu diterbitkan.

`binaryMessage`(Python:) `binary_message`

(Opsional) Sebuah pesan biner. Objek ini, `BinaryMessage`, berisi informasi berikut:

`message`

Pesan biner sebagai gumpalan.

`context`

Konteks pesan, seperti topik di mana pesan itu diterbitkan.

Fitur ini tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass. Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk mengakses konteks pesan.

SDK	Versi minimum	
AWS IoT Device SDK untuk Java v2	v1.9.3	

SDK	Versi minimum
AWS IoT Device SDK untuk Python v2	v1.11.3
AWS IoT Device SDK untuk C++ v2	v1.18.4
AWS IoT Device SDK for JavaScript v2	v1.12.0

> **Note**
>
> Perangkat lunak AWS IoT Greengrass Core menggunakan objek pesan yang sama dalam `PublishToTopic` dan `SubscribeToTopic` operasi. Perangkat lunak AWS IoT Greengrass Core menetapkan objek konteks ini dalam pesan saat Anda berlangganan, dan mengabaikan objek konteks ini dalam pesan yang Anda terbitkan.

Objek ini, `MessageContext`, berisi informasi berikut:

`topic`

Topik di mana pesan itu diterbitkan.

`topicName`(Python:) `topic_name`

Topik yang pesannya dipublikasikan.

> **Note**
>
> Properti ini saat ini tidak digunakan. Di Greengrass nucleus v2.6.0 dan yang lebih baru, Anda bisa mendapatkan nilai dari `SubscriptionResponseMessage` a untuk mendapatkan topik `(jsonMessage|binaryMessage).context.topic` di mana pesan itu diterbitkan.

Contoh

Contoh-contoh berikut ini menunjukkan cara memanggil operasi ini dalam kode komponen kustom.

Java (IPC client V2)

Example Contoh: Berlangganan pesan publikasi/langganan lokal

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.SubscribeToTopicResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.*;

import java.nio.charset.StandardCharsets;
import java.util.Optional;

public class SubscribeToTopicV2 {

    public static void main(String[] args) {
        String topic = args[0];
        try (GreengrassCoreIPCClientV2 ipcClient =
 GreengrassCoreIPCClientV2.builder().build()) {
            SubscribeToTopicRequest request = new
 SubscribeToTopicRequest().withTopic(topic);
            GreengrassCoreIPCClientV2.StreamingResponse<SubscribeToTopicResponse,
                    SubscribeToTopicResponseHandler> response =
                    ipcClient.subscribeToTopic(request,
 SubscribeToTopicV2::onStreamEvent,
                            Optional.of(SubscribeToTopicV2::onStreamError),
                            Optional.of(SubscribeToTopicV2::onStreamClosed));
            SubscribeToTopicResponseHandler responseHandler =
 response.getHandler();
            System.out.println("Successfully subscribed to topic: " + topic);

            // Keep the main thread alive, or the process will exit.
            try {
                while (true) {
                    Thread.sleep(10000);
                }
            } catch (InterruptedException e) {
                System.out.println("Subscribe interrupted.");
            }
```

```
            // To stop subscribing, close the stream.
            responseHandler.closeStream();
        } catch (Exception e) {
            if (e.getCause() instanceof UnauthorizedError) {
                System.err.println("Unauthorized error while publishing to topic: "
 + topic);
            } else {
                System.err.println("Exception occurred when using IPC.");
            }
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static void onStreamEvent(SubscriptionResponseMessage
 subscriptionResponseMessage) {
        try {
            BinaryMessage binaryMessage =
 subscriptionResponseMessage.getBinaryMessage();
            String message = new String(binaryMessage.getMessage(),
 StandardCharsets.UTF_8);
            String topic = binaryMessage.getContext().getTopic();
            System.out.printf("Received new message on topic %s: %s%n", topic,
 message);
        } catch (Exception e) {
            System.err.println("Exception occurred while processing subscription
 response " +
                    "message.");
            e.printStackTrace();
        }
    }

    public static boolean onStreamError(Throwable error) {
        System.err.println("Received a stream error.");
        error.printStackTrace();
        return false; // Return true to close stream, false to keep stream open.
    }

    public static void onStreamClosed() {
        System.out.println("Subscribe to topic stream closed.");
    }
}
```

Python (IPC client V2)

Example Contoh: Berlangganan pesan publikasi/langganan lokal

```
import sys
import time
import traceback

from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
from awsiot.greengrasscoreipc.model import (
    SubscriptionResponseMessage,
    UnauthorizedError
)


def main():
    args = sys.argv[1:]
    topic = args[0]

    try:
        ipc_client = GreengrassCoreIPCClientV2()
        # Subscription operations return a tuple with the response and the
 operation.
        _, operation = ipc_client.subscribe_to_topic(topic=topic,
 on_stream_event=on_stream_event,

 on_stream_error=on_stream_error, on_stream_closed=on_stream_closed)
        print('Successfully subscribed to topic: ' + topic)

        # Keep the main thread alive, or the process will exit.
        try:
            while True:
                time.sleep(10)
        except InterruptedError:
            print('Subscribe interrupted.')

        # To stop subscribing, close the stream.
        operation.close()
    except UnauthorizedError:
        print('Unauthorized error while subscribing to topic: ' +
              topic, file=sys.stderr)
        traceback.print_exc()
        exit(1)
    except Exception:
```

```
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)


def on_stream_event(event: SubscriptionResponseMessage) -> None:
    try:
        message = str(event.binary_message.message, 'utf-8')
        topic = event.binary_message.context.topic
        print('Received new message on topic %s: %s' % (topic, message))
    except:
        traceback.print_exc()


def on_stream_error(error: Exception) -> bool:
    print('Received a stream error.', file=sys.stderr)
    traceback.print_exc()
    return False  # Return True to close stream, False to keep stream open.


def on_stream_closed() -> None:
    print('Subscribe to topic stream closed.')


if __name__ == '__main__':
    main()
```

C++

Example Contoh: Berlangganan pesan publikasi/langganan lokal

```
#include <iostream>

#include </crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class SubscribeResponseHandler : public SubscribeToTopicStreamHandler {
    public:
        virtual ~SubscribeResponseHandler() {}

    private:
```

```
        void OnStreamEvent(SubscriptionResponseMessage *response) override {
            auto jsonMessage = response->GetJsonMessage();
            if (jsonMessage.has_value() &&
 jsonMessage.value().GetMessage().has_value()) {
                auto messageString =
 jsonMessage.value().GetMessage().value().View().WriteReadable();
                // Handle JSON message.
            } else {
                auto binaryMessage = response->GetBinaryMessage();
                if (binaryMessage.has_value() &&
 binaryMessage.value().GetMessage().has_value()) {
                    auto messageBytes = binaryMessage.value().GetMessage().value();
                    std::string messageString(messageBytes.begin(),
 messageBytes.end());
                    // Handle binary message.
                }
            }
        }

        bool OnStreamError(OperationError *error) override {
            // Handle error.
            return false; // Return true to close stream, false to keep stream open.
        }

        void OnStreamClosed() override {
            // Handle close.
        }
};

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
};
```

```
int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String topic("my/topic");
    int timeout = 10;

    SubscribeToTopicRequest request;
    request.SetTopic(topic);

    //SubscribeResponseHandler streamHandler;
    auto streamHandler = MakeShared<SubscribeResponseHandler>(DefaultAllocator());
    auto operation = ipcClient.NewSubscribeToTopic(streamHandler);
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
 Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
        } else {
            // Handle RPC error.
```

```
        }
        exit(-1);
    }

    // Keep the main thread alive, or the process will exit.
    while (true) {
        std::this_thread::sleep_for(std::chrono::seconds(10));
    }

    operation->Close();
    return 0;
}
```

JavaScript

Example Contoh: Berlangganan pesan publikasi/langganan lokal

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
import {SubscribeToTopicRequest, SubscriptionResponseMessage} from "aws-iot-device-
sdk-v2/dist/greengrasscoreipc/model";
import {RpcError} from "aws-iot-device-sdk-v2/dist/eventstream_rpc";

class SubscribeToTopic {
    private ipcClient : greengrasscoreipc.Client
    private readonly topic : string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.subscribeToTopic().then(r => console.log("Started workflow"));
    }

    private async subscribeToTopic() {
        try {
            this.ipcClient = await getIpcClient();

            const subscribeToTopicRequest : SubscribeToTopicRequest = {
                topic: this.topic,
            }

            const streamingOperation =
 this.ipcClient.subscribeToTopic(subscribeToTopicRequest, undefined); //
 conditionally apply options
```

```
            streamingOperation.on("message", (message: SubscriptionResponseMessage)
 => {
                // parse the message depending on your use cases, e.g.
                if(message.binaryMessage && message.binaryMessage.message) {
                    const receivedMessage =
 message.binaryMessage?.message.toString();
                }
            });

            streamingOperation.on("streamError", (error : RpcError) => {
                // define your own error handling logic
            })

            streamingOperation.on("ended", () => {
                // define your own logic
            })

            await streamingOperation.activate();

            // Keep the main thread alive, or the process will exit.
            await new Promise((resolve) => setTimeout(resolve, 10000))
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}

export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}
```

```
// starting point
const subscribeToTopic = new SubscribeToTopic();
```

Contoh

Gunakan contoh berikut untuk mempelajari cara menggunakan layanan IPC publikasi/berlangganan dalam komponen Anda.

Contoh penerbit/berlangganan penerbit (Java, klien IPC V1)

Contoh resep berikut memungkinkan komponen untuk mempublikasikan ke semua topik.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubPublisherJava",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that publishes messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubPublisherJava:pubsub:1": {
            "policyDescription": "Allows access to publish to all topics.",
            "operations": [
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "java -jar {artifacts:path}/PubSubPublisher.jar"
      }
```

```
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubPublisherJava
ComponentVersion: '1.0.0'
ComponentDescription: A component that publishes messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        'com.example.PubSubPublisherJava:pubsub:1':
          policyDescription: Allows access to publish to all topics.
          operations:
            - 'aws.greengrass#PublishToTopic'
          resources:
            - '*'
Manifests:
  - Lifecycle:
      run: |-
        java -jar {artifacts:path}/PubSubPublisher.jar
```

Aplikasi contoh Java berikut ini menunjukkan cara menggunakan layanan IPC publikasi/berlangganan untuk mempublikasikan pesan ke komponen lain.

```
/* Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: Apache-2.0 */

package com.example.ipc.pubsub;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.model.*;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
```

```
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class PubSubPublisher {

    public static void main(String[] args) {
        String message = "Hello from the pub/sub publisher (Java).";
        String topic = "test/topic/java";

        try (EventStreamRPCConnection eventStreamRPCConnection =
 IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient = new
 GreengrassCoreIPCClient(eventStreamRPCConnection);

            while (true) {
                PublishToTopicRequest publishRequest = new PublishToTopicRequest();
                PublishMessage publishMessage = new PublishMessage();
                BinaryMessage binaryMessage = new BinaryMessage();
                binaryMessage.setMessage(message.getBytes(StandardCharsets.UTF_8));
                publishMessage.setBinaryMessage(binaryMessage);
                publishRequest.setPublishMessage(publishMessage);
                publishRequest.setTopic(topic);
                CompletableFuture<PublishToTopicResponse> futureResponse = ipcClient
                        .publishToTopic(publishRequest,
 Optional.empty()).getResponse();

                try {
                    futureResponse.get(10, TimeUnit.SECONDS);
                    System.out.println("Successfully published to topic: " + topic);
                } catch (TimeoutException e) {
                    System.err.println("Timeout occurred while publishing to topic: " +
 topic);
                } catch (ExecutionException e) {
                    if (e.getCause() instanceof UnauthorizedError) {
                        System.err.println("Unauthorized error while publishing to
 topic: " + topic);
                    } else {
                        System.err.println("Execution exception while publishing to
 topic: " + topic);
                    }
                    throw e;
                }
                Thread.sleep(5000);
```

```
            }
        } catch (InterruptedException e) {
            System.out.println("Publisher interrupted.");
        } catch (Exception e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

Contoh menerbitkan/berlangganan pelanggan (Java, klien IPC V1)

Contoh resep berikut memungkinkan komponen untuk berlangganan semua topik.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubSubscriberJava",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that subscribes to messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubSubscriberJava:pubsub:1": {
            "policyDescription": "Allows access to subscribe to all topics.",
            "operations": [
              "aws.greengrass#SubscribeToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "java -jar {artifacts:path}/PubSubSubscriber.jar"
```

```
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubSubscriberJava
ComponentVersion: '1.0.0'
ComponentDescription: A component that subscribes to messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        'com.example.PubSubSubscriberJava:pubsub:1':
          policyDescription: Allows access to subscribe to all topics.
          operations:
            - 'aws.greengrass#SubscribeToTopic'
          resources:
            - '*'
Manifests:
  - Lifecycle:
      run: |-
        java -jar {artifacts:path}/PubSubSubscriber.jar
```

Aplikasi contoh Java berikut ini menunjukkan cara menggunakan layanan IPC publikasi/berlangganan untuk berlangganan pesan ke komponen lain.

```
/* Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: Apache-2.0 */

package com.example.ipc.pubsub;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.SubscribeToTopicResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.SubscribeToTopicRequest;
import software.amazon.awssdk.aws.greengrass.model.SubscribeToTopicResponse;
import software.amazon.awssdk.aws.greengrass.model.SubscriptionResponseMessage;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
```

```
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;
import software.amazon.awssdk.eventstreamrpc.StreamResponseHandler;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class PubSubSubscriber {

    public static void main(String[] args) {
        String topic = "test/topic/java";

        try (EventStreamRPCConnection eventStreamRPCConnection =
 IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient = new
 GreengrassCoreIPCClient(eventStreamRPCConnection);

            SubscribeToTopicRequest subscribeRequest = new SubscribeToTopicRequest();
            subscribeRequest.setTopic(topic);
            SubscribeToTopicResponseHandler operationResponseHandler = ipcClient
                    .subscribeToTopic(subscribeRequest, Optional.of(new
 SubscribeResponseHandler()));
            CompletableFuture<SubscribeToTopicResponse> futureResponse =
 operationResponseHandler.getResponse();

            try {
                futureResponse.get(10, TimeUnit.SECONDS);
                System.out.println("Successfully subscribed to topic: " + topic);
            } catch (TimeoutException e) {
                System.err.println("Timeout occurred while subscribing to topic: " +
 topic);
                throw e;
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.println("Unauthorized error while subscribing to topic:
 " + topic);
                } else {
                    System.err.println("Execution exception while subscribing to topic:
 " + topic);
                }
                throw e;
```

```
            }

            // Keep the main thread alive, or the process will exit.
            try {
                while (true) {
                    Thread.sleep(10000);
                }
            } catch (InterruptedException e) {
                System.out.println("Subscribe interrupted.");
            }
        } catch (Exception e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    private static class SubscribeResponseHandler implements
StreamResponseHandler<SubscriptionResponseMessage> {

        @Override
        public void onStreamEvent(SubscriptionResponseMessage
subscriptionResponseMessage) {
            try {
                String message = new
String(subscriptionResponseMessage.getBinaryMessage()
                        .getMessage(), StandardCharsets.UTF_8);
                System.out.println("Received new message: " + message);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        @Override
        public boolean onStreamError(Throwable error) {
            System.err.println("Received a stream error.");
            error.printStackTrace();
            return false; // Return true to close stream, false to keep stream open.
        }

        @Override
        public void onStreamClosed() {
            System.out.println("Subscribe to topic stream closed.");
        }
```

```
    }
}
```

Contoh penerbit/berlangganan penerbit (Python, klien IPC V1)

Contoh resep berikut memungkinkan komponen untuk mempublikasikan ke semua topik.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubPublisherPython",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that publishes messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubPublisherPython:pubsub:1": {
            "policyDescription": "Allows access to publish to all topics.",
            "operations": [
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk",
        "run": "python3 -u {artifacts:path}/pubsub_publisher.py"
      }
    },
    {
      "Platform": {
```

```
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/pubsub_publisher.py"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubPublisherPython
ComponentVersion: 1.0.0
ComponentDescription: A component that publishes messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        com.example.PubSubPublisherPython:pubsub:1:
          policyDescription: Allows access to publish to all topics.
          operations:
            - aws.greengrass#PublishToTopic
          resources:
            - "*"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: python3 -m pip install --user awsiotsdk
      run: python3 -u {artifacts:path}/pubsub_publisher.py
  - Platform:
      os: windows
    Lifecycle:
      install: py -3 -m pip install --user awsiotsdk
      run: py -3 -u {artifacts:path}/pubsub_publisher.py
```

Aplikasi contoh Python berikut ini menunjukkan cara menggunakan layanan IPC publikasi/berlangganan untuk mempublikasikan pesan ke komponen lain.

```
import concurrent.futures
import sys
import time
import traceback

import awsiot.greengrasscoreipc
from awsiot.greengrasscoreipc.model import (
    PublishToTopicRequest,
    PublishMessage,
    BinaryMessage,
    UnauthorizedError
)


topic = "test/topic/python"
message = "Hello from the pub/sub publisher (Python)."
TIMEOUT = 10

try:
    ipc_client = awsiot.greengrasscoreipc.connect()

    while True:
        request = PublishToTopicRequest()
        request.topic = topic
        publish_message = PublishMessage()
        publish_message.binary_message = BinaryMessage()
        publish_message.binary_message.message = bytes(message, "utf-8")
        request.publish_message = publish_message
        operation = ipc_client.new_publish_to_topic()
        operation.activate(request)
        future_response = operation.get_response()

        try:
            future_response.result(TIMEOUT)
            print('Successfully published to topic: ' + topic)
        except concurrent.futures.TimeoutError:
            print('Timeout occurred while publishing to topic: ' + topic,
 file=sys.stderr)
       except UnauthorizedError as e:
            print('Unauthorized error while publishing to topic: ' + topic,
 file=sys.stderr)
            raise e
       except Exception as e:
```

```
            print('Exception while publishing to topic: ' + topic, file=sys.stderr)
            raise e
        time.sleep(5)
except InterruptedError:
    print('Publisher interrupted.')
except Exception:
    print('Exception occurred when using IPC.', file=sys.stderr)
    traceback.print_exc()
    exit(1)
```

Contoh menerbitkan/berlangganan pelanggan (Python, klien IPC V1)

Contoh resep berikut memungkinkan komponen untuk berlangganan semua topik.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubSubscriberPython",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that subscribes to messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubSubscriberPython:pubsub:1": {
            "policyDescription": "Allows access to subscribe to all topics.",
            "operations": [
              "aws.greengrass#SubscribeToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
```

```
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk",
        "run": "python3 -u {artifacts:path}/pubsub_subscriber.py"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/pubsub_subscriber.py"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubSubscriberPython
ComponentVersion: 1.0.0
ComponentDescription: A component that subscribes to messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        com.example.PubSubSubscriberPython:pubsub:1:
          policyDescription: Allows access to subscribe to all topics.
          operations:
            - aws.greengrass#SubscribeToTopic
          resources:
            - "*"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: python3 -m pip install --user awsiotsdk
      run: python3 -u {artifacts:path}/pubsub_subscriber.py
  - Platform:
      os: windows
```

```
    Lifecycle:
      install: py -3 -m pip install --user awsiotsdk
      run: py -3 -u {artifacts:path}/pubsub_subscriber.py
```

Aplikasi contoh Python berikut ini menunjukkan cara menggunakan layanan IPC publikasi/berlangganan untuk berlangganan pesan ke komponen lain.

```
import concurrent.futures
import sys
import time
import traceback

import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import (
    SubscribeToTopicRequest,
    SubscriptionResponseMessage,
    UnauthorizedError
)

topic = "test/topic/python"
TIMEOUT = 10


class StreamHandler(client.SubscribeToTopicStreamHandler):
    def __init__(self):
        super().__init__()

    def on_stream_event(self, event: SubscriptionResponseMessage) -> None:
        try:
            message = str(event.binary_message.message, "utf-8")
            print("Received new message: " + message)
        except:
            traceback.print_exc()

    def on_stream_error(self, error: Exception) -> bool:
        print("Received a stream error.", file=sys.stderr)
        traceback.print_exc()
        return False  # Return True to close stream, False to keep stream open.

    def on_stream_closed(self) -> None:
        print('Subscribe to topic stream closed.')
```

```
try:
    ipc_client = awsiot.greengrasscoreipc.connect()

    request = SubscribeToTopicRequest()
    request.topic = topic
    handler = StreamHandler()
    operation = ipc_client.new_subscribe_to_topic(handler)
    operation.activate(request)
    future_response = operation.get_response()

    try:
        future_response.result(TIMEOUT)
        print('Successfully subscribed to topic: ' + topic)
    except concurrent.futures.TimeoutError as e:
        print('Timeout occurred while subscribing to topic: ' + topic,
 file=sys.stderr)
        raise e
    except UnauthorizedError as e:
        print('Unauthorized error while subscribing to topic: ' + topic,
 file=sys.stderr)
        raise e
    except Exception as e:
        print('Exception while subscribing to topic: ' + topic, file=sys.stderr)
        raise e

    # Keep the main thread alive, or the process will exit.
    try:
        while True:
            time.sleep(10)
    except InterruptedError:
        print('Subscribe interrupted.')
except Exception:
    print('Exception occurred when using IPC.', file=sys.stderr)
    traceback.print_exc()
    exit(1)
```

Contoh penerbit publikasi/berlangganan (C++)

Contoh resep berikut memungkinkan komponen untuk mempublikasikan ke semua topik.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubPublisherCpp",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that publishes messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubPublisherCpp:pubsub:1": {
            "policyDescription": "Allows access to publish to all topics.",
            "operations": [
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "{artifacts:path}/greengrassv2_pubsub_publisher"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.PubSubPublisherCpp/1.0.0/greengrassv2_pubsub_publisher",
          "Permission": {
            "Execute": "OWNER"
          }
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubPublisherCpp
ComponentVersion: 1.0.0
ComponentDescription: A component that publishes messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        com.example.PubSubPublisherCpp:pubsub:1:
          policyDescription: Allows access to publish to all topics.
          operations:
            - aws.greengrass#PublishToTopic
          resources:
            - "*"
Manifests:
  - Lifecycle:
      run: "{artifacts:path}/greengrassv2_pubsub_publisher"
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.PubSubPublisherCpp/1.0.0/
greengrassv2_pubsub_publisher
        Permission:
          Execute: OWNER
```

Aplikasi contoh C++ berikut ini menunjukkan cara menggunakan layanan IPC publikasi/berlangganan untuk mempublikasikan pesan ke komponen lain.

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        std::cout << "OnConnectCallback" << std::endl;
    }
```

```
    void OnDisconnectCallback(RpcError error) override {
        std::cout << "OnDisconnectCallback: " << error.StatusToString() << std::endl;
        exit(-1);
    }

    bool OnErrorCallback(RpcError error) override {
        std::cout << "OnErrorCallback: " << error.StatusToString() << std::endl;
        return true;
    }
};

int main() {
    String message("Hello from the pub/sub publisher (C++).");
    String topic("test/topic/cpp");
    int timeout = 10;

    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    while (true) {
        PublishToTopicRequest request;
        Vector<uint8_t> messageData({message.begin(), message.end()});
        BinaryMessage binaryMessage;
        binaryMessage.SetMessage(messageData);
        PublishMessage publishMessage;
        publishMessage.SetBinaryMessage(binaryMessage);
        request.SetTopic(topic);
        request.SetPublishMessage(publishMessage);

        auto operation = ipcClient.NewPublishToTopic();
        auto activate = operation->Activate(request, nullptr);
        activate.wait();
```

```
        auto responseFuture = operation->GetResult();
        if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
            std::cerr << "Operation timed out while waiting for response from
 Greengrass Core." << std::endl;
            exit(-1);
        }

        auto response = responseFuture.get();
        if (response) {
            std::cout << "Successfully published to topic: " << topic << std::endl;
        } else {
            // An error occurred.
            std::cout << "Failed to publish to topic: " << topic << std::endl;
            auto errorType = response.GetResultType();
            if (errorType == OPERATION_ERROR) {
                auto *error = response.GetOperationError();
                std::cout << "Operation error: " << error->GetMessage().value() <<
 std::endl;
            } else {
                std::cout << "RPC error: " << response.GetRpcError() << std::endl;
            }
            exit(-1);
        }

        std::this_thread::sleep_for(std::chrono::seconds(5));
    }

    return 0;
}
```

Contoh pelanggan publikasi/berlangganan (C++)

Contoh resep berikut memungkinkan komponen untuk berlangganan semua topik.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubSubscriberCpp",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that subscribes to messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
```

```
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubSubscriberCpp:pubsub:1": {
            "policyDescription": "Allows access to subscribe to all topics.",
            "operations": [
              "aws.greengrass#SubscribeToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "{artifacts:path}/greengrassv2_pub_sub_subscriber"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.PubSubSubscriberCpp/1.0.0/greengrassv2_pub_sub_subscriber",
          "Permission": {
            "Execute": "OWNER"
          }
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubSubscriberCpp
ComponentVersion: 1.0.0
ComponentDescription: A component that subscribes to messages.
ComponentPublisher: Amazon
ComponentConfiguration:
```

```
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        com.example.PubSubSubscriberCpp:pubsub:1:
          policyDescription: Allows access to subscribe to all topics.
          operations:
            - aws.greengrass#SubscribeToTopic
          resources:
            - "*"
Manifests:
  - Lifecycle:
      run: "{artifacts:path}/greengrassv2_pub_sub_subscriber"
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.PubSubSubscriberCpp/1.0.0/greengrassv2_pub_sub_subscriber
        Permission:
          Execute: OWNER
```

Aplikasi contoh C++ berikut ini menunjukkan cara menggunakan layanan IPC publikasi/berlangganan untuk berlangganan pesan ke komponen lain.

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class SubscribeResponseHandler : public SubscribeToTopicStreamHandler {
    public:
        virtual ~SubscribeResponseHandler() {}

    private:
        void OnStreamEvent(SubscriptionResponseMessage *response) override {
            auto jsonMessage = response->GetJsonMessage();
            if (jsonMessage.has_value() &&
 jsonMessage.value().GetMessage().has_value()) {
                auto messageString =
 jsonMessage.value().GetMessage().value().View().WriteReadable();
                std::cout << "Received new message: " << messageString << std::endl;
            } else {
```

```
                auto binaryMessage = response->GetBinaryMessage();
                if (binaryMessage.has_value() &&
 binaryMessage.value().GetMessage().has_value()) {
                    auto messageBytes = binaryMessage.value().GetMessage().value();
                    std::string messageString(messageBytes.begin(),
 messageBytes.end());
                    std::cout << "Received new message: " << messageString <<
 std::endl;
                }
            }
        }

        bool OnStreamError(OperationError *error) override {
            std::cout << "Received an operation error: ";
            if (error->GetMessage().has_value()) {
                std::cout << error->GetMessage().value();
            }
            std::cout << std::endl;
            return false; // Return true to close stream, false to keep stream open.
        }

        void OnStreamClosed() override {
            std::cout << "Subscribe to topic stream closed." << std::endl;
        }
};

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        std::cout << "OnConnectCallback" << std::endl;
    }

    void OnDisconnectCallback(RpcError error) override {
        std::cout << "OnDisconnectCallback: " << error.StatusToString() << std::endl;
        exit(-1);
    }

    bool OnErrorCallback(RpcError error) override {
        std::cout << "OnErrorCallback: " << error.StatusToString() << std::endl;
        return true;
    }
};

int main() {
    String topic("test/topic/cpp");
```

```
    int timeout = 10;

    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    SubscribeToTopicRequest request;
    request.SetTopic(topic);
    auto streamHandler = MakeShared<SubscribeResponseHandler>(DefaultAllocator());
    auto operation = ipcClient.NewSubscribeToTopic(streamHandler);
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (response) {
        std::cout << "Successfully subscribed to topic: " << topic << std::endl;
    } else {
        // An error occurred.
        std::cout << "Failed to subscribe to topic: " << topic << std::endl;
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            std::cout << "Operation error: " << error->GetMessage().value() <<
std::endl;
        } else {
            std::cout << "RPC error: " << response.GetRpcError() << std::endl;
        }
```

```
        exit(-1);
    }

    // Keep the main thread alive, or the process will exit.
    while (true) {
        std::this_thread::sleep_for(std::chrono::seconds(10));
    }

    operation->Close();
    return 0;
}
```

Terbitkan/berlangganan pesan MQTT AWS IoT Core

Layanan AWS IoT Core MQTT messaging IPC memungkinkan Anda mengirim dan menerima pesan MQTT ke dan dari. AWS IoT Core Komponen dapat mempublikasikan pesan ke AWS IoT Core dan berlangganan topik untuk bertindak atas pesan MQTT dari sumber lain. Untuk informasi selengkapnya tentang AWS IoT Core implementasi MQTT, lihat MQTT di Panduan Pengembang.AWS IoT Core

> **Note**
>
> Layanan MQTT messaging IPC ini memungkinkan Anda bertukar pesan dengan. AWS IoT Core Untuk informasi selengkapnya tentang cara bertukar pesan antar komponen, lihat Pesan lokal publikasi/berlangganan.

Topik

- SDK (Versi Minimum)
- Otorisasi
- PublishToIoTCore
- SubscribeToIoTCore
- Contoh

SDK (Versi Minimum)

Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk mempublikasikan dan berlangganan pesan MQTT ke dan dari. AWS IoT Core

SDK	Versi minimum	
AWS IoT Device SDK untuk Java v2	v1.2.10	
AWS IoT Device SDK untuk Python v2	v1.5.3	
AWS IoT Device SDK untuk C ++ v2	v1.17.0	
AWS IoT Device SDK untuk JavaScript v2	v1.12.0	

Otorisasi

Untuk menggunakan pesan AWS IoT Core MQTT dalam komponen kustom, Anda harus menentukan kebijakan otorisasi yang memungkinkan komponen Anda mengirim dan menerima pesan tentang topik. Untuk informasi tentang cara menentukan kebijakan otorisasi, lihat Otorisasi komponen untuk melakukan operasi IPC.

Kebijakan otorisasi untuk pesan AWS IoT Core MQTT memiliki properti berikut.

Pengenal layanan IPC: `aws.greengrass.ipc.mqttproxy`

Operasi	Deskripsi	Sumber daya
`aws.greengrass#PublishToIoTCore`	Memungkinkan komponen untuk mempublikasikan pesan AWS IoT Core pada topik MQTT yang Anda tentukan.	Topik string, seperti `test/topic` , atau * untuk mengizinkan akses ke semua topik. Anda dapat menggunakan wildcard topik MQTT

Operasi	Deskripsi	Sumber daya
		(#dan+) untuk mencocokkan beberapa sumber daya.
`aws.greengrass#SubscribeToIoTCore`	Memungkinkan komponen untuk berlangganan pesan dari AWS IoT Core topik yang Anda tentukan.	Topik string, seperti `test/topic` , atau * untuk mengizinkan akses ke semua topik. Anda dapat menggunakan wildcard topik MQTT (#dan+) untuk mencocokkan beberapa sumber daya.
*	Memungkinkan komponen untuk mempublikasikan dan berlangganan pesan AWS IoT Core MQTT untuk topik yang Anda tentukan.	Topik string, seperti `test/topic` , atau * untuk mengizinkan akses ke semua topik. Anda dapat menggunakan wildcard topik MQTT (#dan+) untuk mencocokkan beberapa sumber daya.

Wildcard MQTT dalam kebijakan otorisasi MQTT AWS IoT Core

Anda dapat menggunakan wildcard MQTT dalam kebijakan otorisasi AWS IoT Core MQTT IPC. Komponen dapat mempublikasikan dan berlangganan topik yang cocok dengan filter topik yang Anda izinkan dalam kebijakan otorisasi. Misalnya, jika kebijakan otorisasi komponen memberikan akses ke`test/topic/#`, komponen dapat berlangganan`test/topic/#`, dan dapat mempublikasikan dan berlangganan. `test/topic/filter`

Variabel resep dalam kebijakan AWS IoT Core otorisasi MQTT

Jika Anda menggunakan v2.6.0 atau yang lebih baru dari inti Greengrass, Anda dapat menggunakan variabel resep dalam kebijakan otorisasi. `{iot:thingName}` Fitur ini memungkinkan Anda mengonfigurasi kebijakan otorisasi tunggal untuk sekelompok perangkat inti, di mana setiap perangkat inti hanya dapat mengakses topik yang berisi namanya sendiri. Misalnya, Anda dapat mengizinkan akses komponen ke sumber daya topik berikut.

```
devices/{iot:thingName}/messages
```

Untuk informasi selengkapnya, lihat Variabel resep dan Gunakan variabel resep dalam menggabungkan pembaruan.

Contoh kebijakan otorisasi

Anda dapat mereferensikan contoh kebijakan otorisasi berikut untuk membantu Anda mengonfigurasi kebijakan otorisasi untuk komponen Anda.

Example Contoh kebijakan otorisasi dengan akses tidak terbatas

Kebijakan otorisasi contoh berikut ini memungkinkan komponen untuk mempublikasikan dan berlangganan semua topik.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "com.example.MyIoTCorePubSubComponent:mqttproxy:1": {
        "policyDescription": "Allows access to publish/subscribe to all topics.",
        "operations": [
          "aws.greengrass#PublishToIoTCore",
          "aws.greengrass#SubscribeToIoTCore"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
}
```

YAML

```
---
accessControl:
  aws.greengrass.ipc.mqttproxy:
    com.example.MyIoTCorePubSubComponent:mqttproxy:1:
      policyDescription: Allows access to publish/subscribe to all topics.
```

```
      operations:
        - aws.greengrass#PublishToIoTCore
        - aws.greengrass#SubscribeToIoTCore
      resources:
        - "*"
```

Example Contoh kebijakan otorisasi dengan akses terbatas

Contoh kebijakan otorisasi berikut memungkinkan komponen untuk menerbitkan dan berlangganan dua topik bernama `factory/1/events` dan`factory/1/actions`.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "com.example.MyIoTCorePubSubComponent:mqttproxy:1": {
        "policyDescription": "Allows access to publish/subscribe to factory 1
 topics.",
        "operations": [
          "aws.greengrass#PublishToIoTCore",
          "aws.greengrass#SubscribeToIoTCore"
        ],
        "resources": [
          "factory/1/actions",
          "factory/1/events"
        ]
      }
    }
  }
}
```

YAML

```
---
accessControl:
  aws.greengrass.ipc.mqttproxy:
    "com.example.MyIoTCorePubSubComponent:mqttproxy:1":
      policyDescription: Allows access to publish/subscribe to factory 1 topics.
      operations:
        - aws.greengrass#PublishToIoTCore
        - aws.greengrass#SubscribeToIoTCore
```

```
      resources:
        - factory/1/actions
        - factory/1/events
```

Example Contoh kebijakan otorisasi untuk sekelompok perangkat inti

> **Important**
>
> Contoh ini menggunakan fitur yang tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass. Greengrass nucleus v2.6.0 menambahkan dukungan untuk sebagian besar variabel resep, seperti, dalam konfigurasi komponen. `{iot:thingName}`

Contoh kebijakan otorisasi berikut memungkinkan komponen untuk menerbitkan dan berlangganan topik yang berisi nama perangkat inti yang menjalankan komponen.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "com.example.MyIoTCorePubSubComponent:mqttproxy:1": {
        "policyDescription": "Allows access to publish/subscribe to all topics.",
        "operations": [
          "aws.greengrass#PublishToIoTCore",
          "aws.greengrass#SubscribeToIoTCore"
        ],
        "resources": [
          "factory/1/devices/{iot:thingName}/controls"
        ]
      }
    }
  }
}
```

YAML

```
---
accessControl:
  aws.greengrass.ipc.mqttproxy:
    "com.example.MyIoTCorePubSubComponent:mqttproxy:1":
```

```
      policyDescription: Allows access to publish/subscribe to all topics.
      operations:
        - aws.greengrass#PublishToIoTCore
        - aws.greengrass#SubscribeToIoTCore
      resources:
        - factory/1/devices/{iot:thingName}/controls
```

PublishToIoTCore

Menerbitkan pesan MQTT ke AWS IoT Core topik.

Saat Anda mempublikasikan pesan MQTT ke AWS IoT Core, ada kuota 100 transaksi per detik. Jika Anda melebihi kuota ini, pesan akan diantrian untuk diproses di perangkat Greengrass. Ada juga kuota 512 Kb data per detik dan kuota akun sebesar 20.000 publikasi per detik (2.000 di beberapa). Wilayah AWS Untuk informasi selengkapnya tentang batas broker pesan MQTT AWS IoT Core, lihat batas dan AWS IoT Core kuota broker pesan dan protokol.

Jika Anda melebihi kuota ini, perangkat Greengrass membatasi penerbitan pesan. AWS IoT Core Pesan disimpan dalam spooler di memori. Secara default, memori yang dialokasikan ke spooler adalah 2,5 Mb. Jika spooler terisi, pesan baru ditolak. Anda dapat meningkatkan ukuran spooler. Untuk informasi lebih lanjut, lihat Konfigurasi di Inti Greengrass dokumentasi. Untuk menghindari pengisian spooler dan perlu menambah memori yang dialokasikan, batasi permintaan publikasi tidak lebih dari 100 permintaan per detik.

Ketika aplikasi Anda perlu mengirim pesan pada tingkat yang lebih tinggi, atau pesan yang lebih besar, pertimbangkan untuk menggunakan pesan Manajer pengaliran untuk mengirim pesan ke Kinesis Data Streams. Komponen manajer aliran dirancang untuk mentransfer data volume tinggi ke file. AWS Cloud Untuk informasi selengkapnya, lihat Kelola aliran data di perangkat inti Greengrass.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`topicName`(Python:) `topic_name`

Topik untuk mempublikasikan pesan.

`qos`

QoS MQTT yang akan digunakan. Enum ini, `QOS`, memiliki nilai-nilai berikut:

- `AT_MOST_ONCE` – QoS 0. Pesan MQTT dikirim paling banyak sekali.
- `AT_LEAST_ONCE` – QoS 1. Pesan MQTT dikirim paling sedikit sekali.

`payload`

(Opsional) Pesan muatan sebagai gumpalan.

Fitur-fitur berikut tersedia untuk v2.10.0 dan yang lebih baru Inti Greengrass saat menggunakan MQTT 5. Fitur-fitur ini diabaikan saat Anda menggunakan MQTT 3.1.1. Tabel berikut mencantumkan versi minimum SDK AWS IoT perangkat yang harus Anda gunakan untuk mengakses fitur ini.

SDK	Versi minimum
AWS IoT Device SDK for Python v2	v1.15.0
AWS IoT Device SDK for Java v2	v1.13.0
AWS IoT Device SDK for C++ v2	v1.24.0
AWS IoT Device SDK for JavaScript v2	v1.13.0

`payloadFormat`

(Opsional) Format payload pesan. Jika Anda tidak menyetel`payloadFormat`, tipenya diasumsikan`BYTES`. Enum memiliki nilai-nilai berikut:

- `BYTES`— Isi muatan adalah gumpalan biner.
- `UTF8`— Isi payload adalah string karakter UTF8.

`retain`

(Opsional) Menunjukkan apakah akan menyetel opsi penyimpanan MQTT saat menerbitkan. `true`

`userProperties`

(Opsional) Daftar `UserProperty` objek khusus aplikasi untuk dikirim. `UserProperty`Objek didefinisikan sebagai berikut:

```
UserProperty:
```

```
  key: string
  value: string
```

`messageExpiryIntervalSeconds`

(Opsional) Jumlah detik sebelum pesan berakhir dan dihapus oleh server. Jika nilai ini tidak disetel, pesan tidak kedaluwarsa.

`correlationData`

(Opsional) Informasi ditambahkan ke permintaan yang dapat digunakan untuk mengaitkan permintaan dengan respons.

`responseTopic`

(Opsional) Topik yang harus digunakan untuk pesan respons.

`contentType`

(Opsional) Pengidentifikasi khusus aplikasi dari jenis konten pesan.

Respons

Operasi ini tidak memberikan informasi apa pun dalam tanggapannya.

Contoh

Contoh-contoh berikut ini menunjukkan cara memanggil operasi ini dalam kode komponen kustom.

Java (IPC client V2)

Example Contoh: Publikasikan pesan

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.model.PublishToIoTCoreRequest;
import software.amazon.awssdk.aws.greengrass.model.QOS;
import java.nio.charset.StandardCharsets;

public class PublishToIoTCore {
```

```
    public static void main(String[] args) {
        String topic = args[0];
        String message = args[1];
        QOS qos = QOS.get(args[2]);

        try (GreengrassCoreIPCClientV2 ipcClientV2 =
 GreengrassCoreIPCClientV2.builder().build()) {
            ipcClientV2.publishToIoTCore(new PublishToIoTCoreRequest()
                    .withTopicName(topic)
                    .withPayload(message.getBytes(StandardCharsets.UTF_8))
                    .withQos(qos));
            System.out.println("Successfully published to topic: " + topic);
        } catch (Exception e) {
            System.err.println("Exception occurred.");
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

Python (IPC client V2)

Example Contoh: Publikasikan pesan

> **Note**
>
> Contoh ini mengasumsikan bahwa Anda menggunakan versi 1.5.4 atau yang lebih baru untuk AWS IoT Device SDK Python v2.

```
import awsiot.greengrasscoreipc.clientv2 as clientV2

topic = 'my/topic'
qos = '1'
payload = 'Hello, World'

ipc_client = clientV2.GreengrassCoreIPCClientV2()
resp = ipc_client.publish_to_iot_core(topic_name=topic, qos=qos, payload=payload)
ipc_client.close()
```

Java (IPC client V1)

Example Contoh: Publikasikan pesan

> **Note**
>
> Contoh ini menggunakan `IPCUtils` kelas untuk membuat koneksi ke layanan AWS IoT Greengrass Core IPC. Untuk informasi selengkapnya, lihat Connect ke layanan AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.PublishToIoTCoreResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.PublishToIoTCoreRequest;
import software.amazon.awssdk.aws.greengrass.model.PublishToIoTCoreResponse;
import software.amazon.awssdk.aws.greengrass.model.QOS;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class PublishToIoTCore {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        String topic = args[0];
        String message = args[1];
        QOS qos = QOS.get(args[2]);
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            PublishToIoTCoreResponseHandler responseHandler =
```

```
                    PublishToIoTCore.publishBinaryMessageToTopic(ipcClient, topic,
 message, qos);
            CompletableFuture<PublishToIoTCoreResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                System.out.println("Successfully published to topic: " + topic);
            } catch (TimeoutException e) {
                System.err.println("Timeout occurred while publishing to topic: " +
 topic);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.println("Unauthorized error while publishing to
 topic: " + topic);
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static PublishToIoTCoreResponseHandler
 publishBinaryMessageToTopic(GreengrassCoreIPCClient greengrassCoreIPCClient, String
 topic, String message, QOS qos) {
        PublishToIoTCoreRequest publishToIoTCoreRequest = new
 PublishToIoTCoreRequest();
        publishToIoTCoreRequest.setTopicName(topic);

 publishToIoTCoreRequest.setPayload(message.getBytes(StandardCharsets.UTF_8));
        publishToIoTCoreRequest.setQos(qos);
        return greengrassCoreIPCClient.publishToIoTCore(publishToIoTCoreRequest,
 Optional.empty());
    }
}
```

Python (IPC client V1)

Example Contoh: Publikasikan pesan

> **Note**
>
> Contoh ini mengasumsikan bahwa Anda menggunakan versi 1.5.4 atau yang lebih baru untuk AWS IoT Device SDK Python v2.

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import (
    QOS,
    PublishToIoTCoreRequest
)

TIMEOUT = 10

ipc_client = awsiot.greengrasscoreipc.connect()

topic = "my/topic"
message = "Hello, World"
qos = QOS.AT_LEAST_ONCE

request = PublishToIoTCoreRequest()
request.topic_name = topic
request.payload = bytes(message, "utf-8")
request.qos = qos
operation = ipc_client.new_publish_to_iot_core()
operation.activate(request)
future_response = operation.get_response()
future_response.result(TIMEOUT)
```

C++

Example Contoh: Publikasikan pesan

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>
```

```
using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
};

int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String message("Hello, World!");
    String topic("my/topic");
    QOS qos = QOS_AT_MOST_ONCE;
    int timeout = 10;

    PublishToIoTCoreRequest request;
    Vector<uint8_t> messageData({message.begin(), message.end()});
    request.SetTopicName(topic);
    request.SetPayload(messageData);
    request.SetQos(qos);

    auto operation = ipcClient.NewPublishToIoTCore();
```

```
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
 Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
        } else {
            // Handle RPC error.
        }
    }

    return 0;
}
```

JavaScript

Example Contoh: Publikasikan pesan

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
import {QOS, PublishToIoTCoreRequest} from "aws-iot-device-sdk-v2/dist/
greengrasscoreipc/model";

class PublishToIoTCore {
    private ipcClient: greengrasscoreipc.Client
    private readonly topic: string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.publishToIoTCore().then(r => console.log("Started workflow"));
```

```
    }

    private async publishToIoTCore() {
        try {
            const request: PublishToIoTCoreRequest = {
                topicName: this.topic,
                qos: QOS.AT_LEAST_ONCE, // you can change this depending on your use
 case
            }

            this.ipcClient = await getIpcClient();

            await this.ipcClient.publishToIoTCore(request);
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}


export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

// starting point
const publishToIoTCore = new PublishToIoTCore();
```

SubscribeToIoTCore

Berlangganan pesan MQTT dari AWS IoT Core topik atau filter topik. Perangkat lunak AWS IoT Greengrass Core menghapus langganan saat komponen mencapai akhir siklus hidupnya.

Operasi ini adalah operasi berlangganan di mana Anda berlangganan aliran pesan peristiwa. Untuk menggunakan operasi ini, tentukan bagian yang menangani respons aliran dengan fungsi yang menangani pesan peristiwa, kesalahan, dan penutupan aliran. Untuk informasi selengkapnya, lihat Berlangganan pengaliran peristiwa IPC.

Jenis pesan peristiwa: `IoTCoreMessage`

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`topicName`(Python:) `topic_name`

Topik yang harus dijadikan langganan. Anda dapat menggunakan wildcard topik MQTT (# dan +) untuk berlangganan beberapa topik.

`qos`

QoS MQTT yang akan digunakan. Enum ini, `QOS`, memiliki nilai-nilai berikut:

- `AT_MOST_ONCE` – QoS 0. Pesan MQTT dikirim paling banyak sekali.
- `AT_LEAST_ONCE` – QoS 1. Pesan MQTT dikirim paling sedikit sekali.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`messages`

Aliran pesan MQTT. Objek ini, `IoTCoreMessage`, berisi informasi berikut:

`message`

Pesan MQTT. Objek ini, `MQTTMessage`, berisi informasi berikut:

`topicName`(Python:) `topic_name`

Topik yang pesannya dipublikasikan.

`payload`

(Opsional) Pesan muatan sebagai gumpalan.

Fitur-fitur berikut tersedia untuk v2.10.0 dan yang lebih baru Inti Greengrass saat menggunakan MQTT 5. Fitur-fitur ini diabaikan saat Anda menggunakan MQTT 3.1.1. Tabel berikut mencantumkan versi minimum SDK AWS IoT perangkat yang harus Anda gunakan untuk mengakses fitur ini.

SDK	Versi minimum
AWS IoT Device SDK for Python v2	v1.15.0
AWS IoT Device SDK for Java v2	v1.13.0
AWS IoT Device SDK for C++ v2	v1.24.0
AWS IoT Device SDK for JavaScript v2	v1.13.0

`payloadFormat`

(Opsional) Format payload pesan. Jika Anda tidak menyetel`payloadFormat`, tipenya diasumsikan`BYTES`. Enum memiliki nilai-nilai berikut:

- `BYTES`— Isi muatan adalah gumpalan biner.
- `UTF8`— Isi payload adalah string karakter UTF8.

`retain`

(Opsional) Menunjukkan apakah akan menyetel opsi penyimpanan MQTT saat menerbitkan. `true`

`userProperties`

(Opsional) Daftar `UserProperty` objek khusus aplikasi untuk dikirim. `UserProperty`Objek didefinisikan sebagai berikut:

```
UserProperty:
  key: string
  value: string
```

`messageExpiryIntervalSeconds`

(Opsional) Jumlah detik sebelum pesan berakhir dan dihapus oleh server. Jika nilai ini tidak disetel, pesan tidak kedaluwarsa.

`correlationData`

(Opsional) Informasi ditambahkan ke permintaan yang dapat digunakan untuk mengaitkan permintaan dengan respons.

`responseTopic`

(Opsional) Topik yang harus digunakan untuk pesan respons.

`contentType`

(Opsional) Pengidentifikasi khusus aplikasi dari jenis konten pesan.

Contoh

Contoh-contoh berikut ini menunjukkan cara memanggil operasi ini dalam kode komponen kustom.

Java (IPC client V2)

Example Contoh: Berlangganan pesan

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.SubscribeToIoTCoreResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.QOS;
import software.amazon.awssdk.aws.greengrass.model.IoTCoreMessage;
import software.amazon.awssdk.aws.greengrass.model.SubscribeToIoTCoreRequest;
import software.amazon.awssdk.aws.greengrass.model.SubscribeToIoTCoreResponse;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.function.Consumer;
import java.util.function.Function;


public class SubscribeToIoTCore {

    public static void main(String[] args) {
        String topic = args[0];
```

```
        QOS qos = QOS.get(args[1]);

        Consumer<IoTCoreMessage> onStreamEvent = ioTCoreMessage ->
                System.out.printf("Received new message on topic %s: %s%n",
                        ioTCoreMessage.getMessage().getTopicName(),
                        new String(ioTCoreMessage.getMessage().getPayload(),
StandardCharsets.UTF_8));

        Optional<Function<Throwable, Boolean>> onStreamError =
                Optional.of(e -> {
                    System.err.println("Received a stream error.");
                    e.printStackTrace();
                    return false;
                });

        Optional<Runnable> onStreamClosed = Optional.of(() ->
                System.out.println("Subscribe to IoT Core stream closed."));

        try (GreengrassCoreIPCClientV2 ipcClientV2 =
GreengrassCoreIPCClientV2.builder().build()) {
            SubscribeToIoTCoreRequest request = new SubscribeToIoTCoreRequest()
                    .withTopicName(topic)
                    .withQos(qos);

            GreengrassCoreIPCClientV2.StreamingResponse<SubscribeToIoTCoreResponse,
SubscribeToIoTCoreResponseHandler>
                    streamingResponse = ipcClientV2.subscribeToIoTCore(request,
onStreamEvent, onStreamError, onStreamClosed);

            streamingResponse.getResponse();
            System.out.println("Successfully subscribed to topic: " + topic);

            // Keep the main thread alive, or the process will exit.
            while (true) {
                Thread.sleep(10000);
            }

            // To stop subscribing, close the stream.
            streamingResponse.getHandler().closeStream();
        } catch (InterruptedException e) {
            System.out.println("Subscribe interrupted.");
        } catch (Exception e) {
            System.err.println("Exception occurred.");
            e.printStackTrace();
```

```
            System.exit(1);
        }
    }
}
```

Python (IPC client V2)

Example Contoh: berlangganan pesan

> **Note**
>
> Contoh ini mengasumsikan bahwa Anda menggunakan versi 1.5.4 atau yang lebih baru untuk AWS IoT Device SDK Python v2.

```
import threading
import traceback

import awsiot.greengrasscoreipc.clientv2 as clientV2

topic = 'my/topic'
qos = '1'

def on_stream_event(event):
    try:
        topic_name = event.message.topic_name
        message = str(event.message.payload, 'utf-8')
        print(f'Received new message on topic {topic_name}:  {message}')
    except:
        traceback.print_exc()

def on_stream_error(error):
    # Return True to close stream, False to keep stream open.
    return True

def on_stream_closed():
    pass

ipc_client = clientV2.GreengrassCoreIPCClientV2()
resp, operation = ipc_client.subscribe_to_iot_core(
    topic_name=topic,
    qos=qos,
```

```
    on_stream_event=on_stream_event,
    on_stream_error=on_stream_error,
    on_stream_closed=on_stream_closed
)

# Keep the main thread alive, or the process will exit.
event = threading.Event()
event.wait()

# To stop subscribing, close the operation stream.
operation.close()
ipc_client.close()
```

Java (IPC client V1)

Example Contoh: Berlangganan pesan

> **Note**
>
> Contoh ini menggunakan `IPCUtils` kelas untuk membuat koneksi ke layanan AWS IoT Greengrass Core IPC. Untuk informasi selengkapnya, lihat Connect ke layanan AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.SubscribeToIoTCoreResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.*;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;
import software.amazon.awssdk.eventstreamrpc.StreamResponseHandler;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class SubscribeToIoTCore {
```

```
    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        String topic = args[0];
        QOS qos = QOS.get(args[1]);
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            StreamResponseHandler<IoTCoreMessage> streamResponseHandler =
                    new SubscriptionResponseHandler();
            SubscribeToIoTCoreResponseHandler responseHandler =
                    SubscribeToIoTCore.subscribeToIoTCore(ipcClient, topic, qos,
                            streamResponseHandler);
            CompletableFuture<SubscribeToIoTCoreResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                System.out.println("Successfully subscribed to topic: " + topic);
            } catch (TimeoutException e) {
                System.err.println("Timeout occurred while subscribing to topic: " +
topic);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.println("Unauthorized error while subscribing to
topic: " + topic);
                } else {
                    throw e;
                }
            }

            // Keep the main thread alive, or the process will exit.
            try {
                while (true) {
                    Thread.sleep(10000);
                }
            } catch (InterruptedException e) {
                System.out.println("Subscribe interrupted.");
            }

            // To stop subscribing, close the stream.
            responseHandler.closeStream();
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
```

```
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static SubscribeToIoTCoreResponseHandler
subscribeToIoTCore(GreengrassCoreIPCClient greengrassCoreIPCClient, String topic,
QOS qos, StreamResponseHandler<IoTCoreMessage> streamResponseHandler) {
        SubscribeToIoTCoreRequest subscribeToIoTCoreRequest = new
SubscribeToIoTCoreRequest();
        subscribeToIoTCoreRequest.setTopicName(topic);
        subscribeToIoTCoreRequest.setQos(qos);
        return
greengrassCoreIPCClient.subscribeToIoTCore(subscribeToIoTCoreRequest,
                Optional.of(streamResponseHandler));
    }

    public static class SubscriptionResponseHandler implements
StreamResponseHandler<IoTCoreMessage> {

        @Override
        public void onStreamEvent(IoTCoreMessage ioTCoreMessage) {
            try {
                String topic = ioTCoreMessage.getMessage().getTopicName();
                String message = new
String(ioTCoreMessage.getMessage().getPayload(),
                        StandardCharsets.UTF_8);
                System.out.printf("Received new message on topic %s: %s%n", topic,
message);
            } catch (Exception e) {
                System.err.println("Exception occurred while processing subscription
response " +
                        "message.");
                e.printStackTrace();
            }
        }

        @Override
        public boolean onStreamError(Throwable error) {
            System.err.println("Received a stream error.");
            error.printStackTrace();
            return false;
```

```
        }

        @Override
        public void onStreamClosed() {
            System.out.println("Subscribe to IoT Core stream closed.");
        }
    }
}
```

Python (IPC client V1)

Example Contoh: Berlangganan pesan

> **Note**
>
> Contoh ini mengasumsikan bahwa Anda menggunakan versi 1.5.4 atau yang lebih baru untuk AWS IoT Device SDK Python v2.

```
import time
import traceback

import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import (
    IoTCoreMessage,
    QOS,
    SubscribeToIoTCoreRequest
)

TIMEOUT = 10

ipc_client = awsiot.greengrasscoreipc.connect()

class StreamHandler(client.SubscribeToIoTCoreStreamHandler):
    def __init__(self):
        super().__init__()

    def on_stream_event(self, event: IoTCoreMessage) -> None:
        try:
            message = str(event.message.payload, "utf-8")
            topic_name = event.message.topic_name
            # Handle message.
```

```
        except:
            traceback.print_exc()

    def on_stream_error(self, error: Exception) -> bool:
        # Handle error.
        return True  # Return True to close stream, False to keep stream open.

    def on_stream_closed(self) -> None:
        # Handle close.
        pass


topic = "my/topic"
qos = QOS.AT_MOST_ONCE

request = SubscribeToIoTCoreRequest()
request.topic_name = topic
request.qos = qos
handler = StreamHandler()
operation = ipc_client.new_subscribe_to_iot_core(handler)
operation.activate(request)
future_response = operation.get_response()
future_response.result(TIMEOUT)

# Keep the main thread alive, or the process will exit.
while True:
    time.sleep(10)

# To stop subscribing, close the operation stream.
operation.close()
```

C++

Example Contoh: Berlangganan pesan

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IoTCoreResponseHandler : public SubscribeToIoTCoreStreamHandler {
```

```
    public:
        virtual ~IoTCoreResponseHandler() {}

    private:
        void OnStreamEvent(IoTCoreMessage *response) override {
            auto message = response->GetMessage();
            if (message.has_value() && message.value().GetPayload().has_value()) {
                auto messageBytes = message.value().GetPayload().value();
                std::string messageString(messageBytes.begin(), messageBytes.end());
                std::string topicName =
 message.value().GetTopicName().value().c_str();
                // Handle message.
            }
        }

        bool OnStreamError(OperationError *error) override {
            // Handle error.
            return false; // Return true to close stream, false to keep stream open.
        }

        void OnStreamClosed() override {
            // Handle close.
        }
};

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
};

int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
```

```
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String topic("my/topic");
    QOS qos = QOS_AT_MOST_ONCE;
    int timeout = 10;

    SubscribeToIoTCoreRequest request;
    request.SetTopicName(topic);
    request.SetQos(qos);
    auto streamHandler = MakeShared<IoTCoreResponseHandler>(DefaultAllocator());
    auto operation = ipcClient.NewSubscribeToIoTCore(streamHandler);
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
        } else {
            // Handle RPC error.
        }
        exit(-1);
    }
```

```
    // Keep the main thread alive, or the process will exit.
    while (true) {
        std::this_thread::sleep_for(std::chrono::seconds(10));
    }

    operation->Close();
    return 0;
}
```

JavaScript

Example Contoh: Berlangganan pesan

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
import {IoTCoreMessage, QOS, SubscribeToIoTCoreRequest} from "aws-iot-device-sdk-v2/
dist/greengrasscoreipc/model";
import {RpcError} from "aws-iot-device-sdk-v2/dist/eventstream_rpc";

class SubscribeToIoTCore {
    private ipcClient: greengrasscoreipc.Client
    private readonly topic: string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.subscribeToIoTCore().then(r => console.log("Started workflow"));
    }

    private async subscribeToIoTCore() {
        try {
            const request: SubscribeToIoTCoreRequest = {
                topicName: this.topic,
                qos: QOS.AT_LEAST_ONCE, // you can change this depending on your use
 case
            }

            this.ipcClient = await getIpcClient();

            const streamingOperation = this.ipcClient.subscribeToIoTCore(request);

            streamingOperation.on('message', (message: IoTCoreMessage) => {
                // parse the message depending on your use cases, e.g.
```

```
                if (message.message && message.message.payload) {
                    const receivedMessage = message.message.payload.toString();
                }
            });

            streamingOperation.on('streamError', (error : RpcError) => {
                // define your own error handling logic
            });

            streamingOperation.on('ended', () => {
                // define your own logic
            });

            await streamingOperation.activate();

            // Keep the main thread alive, or the process will exit.
            await new Promise((resolve) => setTimeout(resolve, 10000))
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}

export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

// starting point
const subscribeToIoTCore = new SubscribeToIoTCore();
```

Contoh

Gunakan contoh berikut untuk mempelajari cara menggunakan layanan AWS IoT Core MQTT IPC di komponen Anda.

Contoh penerbit AWS IoT Core MQTT (C++)

Contoh resep berikut memungkinkan komponen untuk mempublikasikan ke semua topik.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.IoTCorePublisherCpp",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that publishes MQTT messages to IoT Core.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.mqttproxy": {
          "com.example.IoTCorePublisherCpp:mqttproxy:1": {
            "policyDescription": "Allows access to publish to all topics.",
            "operations": [
              "aws.greengrass#PublishToIoTCore"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "{artifacts:path}/greengrassv2_iotcore_publisher"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.IoTCorePublisherCpp/1.0.0/greengrassv2_iotcore_publisher",
```

```
          "Permission": {
            "Execute": "OWNER"
          }
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.IoTCorePublisherCpp
ComponentVersion: 1.0.0
ComponentDescription: A component that publishes MQTT messages to IoT Core.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.mqttproxy:
        com.example.IoTCorePublisherCpp:mqttproxy:1:
          policyDescription: Allows access to publish to all topics.
          operations:
            - aws.greengrass#PublishToIoTCore
          resources:
            - "*"
Manifests:
  - Lifecycle:
      run: "{artifacts:path}/greengrassv2_iotcore_publisher"
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.IoTCorePublisherCpp/1.0.0/greengrassv2_iotcore_publisher
        Permission:
          Execute: OWNER
```

Contoh berikut aplikasi C++ menunjukkan bagaimana menggunakan layanan AWS IoT Core MQTT IPC untuk mempublikasikan pesan ke. AWS IoT Core

```
#include <iostream>

#include <aws/crt/Api.h>
```

```
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        std::cout << "OnConnectCallback" << std::endl;
    }

    void OnDisconnectCallback(RpcError error) override {
        std::cout << "OnDisconnectCallback: " << error.StatusToString() << std::endl;
        exit(-1);
    }

    bool OnErrorCallback(RpcError error) override {
        std::cout << "OnErrorCallback: " << error.StatusToString() << std::endl;
        return true;
    }
};

int main() {
    String message("Hello from the Greengrass IPC MQTT publisher (C++).");
    String topic("test/topic/cpp");
    QOS qos = QOS_AT_LEAST_ONCE;
    int timeout = 10;

    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    while (true) {
        PublishToIoTCoreRequest request;
        Vector<uint8_t> messageData({message.begin(), message.end()});
        request.SetTopicName(topic);
```

```
        request.SetPayload(messageData);
        request.SetQos(qos);

        auto operation = ipcClient.NewPublishToIoTCore();
        auto activate = operation->Activate(request, nullptr);
        activate.wait();

        auto responseFuture = operation->GetResult();
        if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
            std::cerr << "Operation timed out while waiting for response from
 Greengrass Core." << std::endl;
            exit(-1);
        }

        auto response = responseFuture.get();
        if (response) {
            std::cout << "Successfully published to topic: " << topic << std::endl;
        } else {
            // An error occurred.
            std::cout << "Failed to publish to topic: " << topic << std::endl;
            auto errorType = response.GetResultType();
            if (errorType == OPERATION_ERROR) {
                auto *error = response.GetOperationError();
                std::cout << "Operation error: " << error->GetMessage().value() <<
 std::endl;
            } else {
                std::cout << "RPC error: " << response.GetRpcError() << std::endl;
            }
            exit(-1);
        }

        std::this_thread::sleep_for(std::chrono::seconds(5));
    }

    return 0;
}
```

Contoh pelanggan AWS IoT Core MQTT (C ++)

Contoh resep berikut memungkinkan komponen untuk berlangganan semua topik.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.IoTCoreSubscriberCpp",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that subscribes to MQTT messages from IoT
 Core.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.mqttproxy": {
          "com.example.IoTCoreSubscriberCpp:mqttproxy:1": {
            "policyDescription": "Allows access to subscribe to all topics.",
            "operations": [
              "aws.greengrass#SubscribeToIoTCore"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "{artifacts:path}/greengrassv2_iotcore_subscriber"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.IoTCoreSubscriberCpp/1.0.0/greengrassv2_iotcore_subscriber",
          "Permission": {
            "Execute": "OWNER"
          }
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.IoTCoreSubscriberCpp
ComponentVersion: 1.0.0
ComponentDescription: A component that subscribes to MQTT messages from IoT Core.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.mqttproxy:
        com.example.IoTCoreSubscriberCpp:mqttproxy:1:
          policyDescription: Allows access to subscribe to all topics.
          operations:
            - aws.greengrass#SubscribeToIoTCore
          resources:
            - "*"
Manifests:
  - Lifecycle:
      run: "{artifacts:path}/greengrassv2_iotcore_subscriber"
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.IoTCoreSubscriberCpp/1.0.0/greengrassv2_iotcore_subscriber
        Permission:
          Execute: OWNER
```

Contoh berikut aplikasi C++ menunjukkan bagaimana menggunakan layanan AWS IoT Core MQTT IPC untuk berlangganan pesan dari. AWS IoT Core

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IoTCoreResponseHandler : public SubscribeToIoTCoreStreamHandler {

    public:
        virtual ~IoTCoreResponseHandler() {}
```

```
    private:

        void OnStreamEvent(IoTCoreMessage *response) override {
            auto message = response->GetMessage();
            if (message.has_value() && message.value().GetPayload().has_value()) {
                auto messageBytes = message.value().GetPayload().value();
                std::string messageString(messageBytes.begin(), messageBytes.end());
                std::string messageTopic =
 message.value().GetTopicName().value().c_str();
                std::cout << "Received new message on topic: " << messageTopic <<
 std::endl;
                std::cout << "Message: " << messageString << std::endl;
            }
        }

        bool OnStreamError(OperationError *error) override {
            std::cout << "Received an operation error: ";
            if (error->GetMessage().has_value()) {
                std::cout << error->GetMessage().value();
            }
            std::cout << std::endl;
            return false; // Return true to close stream, false to keep stream open.
        }

        void OnStreamClosed() override {
            std::cout << "Subscribe to IoT Core stream closed." << std::endl;
        }
};

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        std::cout << "OnConnectCallback" << std::endl;
    }

    void OnDisconnectCallback(RpcError error) override {
        std::cout << "OnDisconnectCallback: " << error.StatusToString() << std::endl;
        exit(-1);
    }

    bool OnErrorCallback(RpcError error) override {
        std::cout << "OnErrorCallback: " << error.StatusToString() << std::endl;
        return true;
    }
```

```
};

int main() {
    String topic("test/topic/cpp");
    QOS qos = QOS_AT_LEAST_ONCE;
    int timeout = 10;

    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    SubscribeToIoTCoreRequest request;
    request.SetTopicName(topic);
    request.SetQos(qos);
    auto streamHandler = MakeShared<IoTCoreResponseHandler>(DefaultAllocator());
    auto operation = ipcClient.NewSubscribeToIoTCore(streamHandler);
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
 Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (response) {
        std::cout << "Successfully subscribed to topic: " << topic << std::endl;
    } else {
        // An error occurred.
        std::cout << "Failed to subscribe to topic: " << topic << std::endl;
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
```

```
            auto *error = response.GetOperationError();
            std::cout << "Operation error: " << error->GetMessage().value() <<
 std::endl;
        } else {
            std::cout << "RPC error: " << response.GetRpcError() << std::endl;
        }
        exit(-1);
    }

    // Keep the main thread alive, or the process will exit.
    while (true) {
        std::this_thread::sleep_for(std::chrono::seconds(10));
    }

    operation->Close();
    return 0;
}
```

Berinteraksilah dengan siklus hidup komponen

Gunakan layanan IPC siklus hidup komponen untuk:

- Perbarui status komponen pada perangkat inti.
- Berlangganan pembaruan status komponen.
- Mencegah nukleus menghentikan komponen untuk menerapkan pembaruan selama penerapan.
- Jeda dan lanjutkan proses komponen.

Topik

- SDK (Versi Minimum)
- Otorisasi
- UpdateState
- SubscribeToComponentUpdates
- DeferComponentUpdate
- PauseComponent
- ResumeComponent

SDK (Versi Minimum)

Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk berinteraksi dengan siklus hidup komponen.

SDK	Versi minimum	
AWS IoT Device SDKuntuk Java v2	v1.2.10	
AWS IoT Device SDKuntuk Python v2	v1.5.3	
AWS IoT Device SDKuntuk C++ v2	v1.17.0	
AWS IoT Device SDKuntuk JavaScript v2	v1.12.0	

Otorisasi

Untuk menjeda atau melanjutkan komponen lain dari komponen kustom, Anda harus menentukan kebijakan otorisasi yang memungkinkan komponen Anda mengelola komponen lain. Untuk informasi tentang cara menentukan kebijakan otorisasi, lihat Otorisasi komponen untuk melakukan operasi IPC.

Kebijakan otorisasi untuk manajemen siklus hidup komponen memiliki properti berikut.

Pengenal layanan IPC: `aws.greengrass.ipc.lifecycle`

Operasi	Deskripsi	Sumber daya
`aws.greengrass#PauseComponent`	Memungkinkan komponen untuk menjeda komponen yang Anda tentukan.	Nama komponen, atau * untuk mengizinkan akses ke semua komponen.
`aws.greengrass#ResumeComponent`	Memungkinkan komponen untuk melanjutkan komponen yang Anda tentukan.	Nama komponen, atau * untuk mengizinkan akses ke semua komponen.

Operasi	Deskripsi	Sumber daya
*	Memungkinkan komponen untuk menjeda dan melanjutkan komponen yang Anda tentukan.	Nama komponen, atau * untuk mengizinkan akses ke semua komponen.

Contoh kebijakan otorisasi

Anda dapat mereferensikan contoh kebijakan otorisasi berikut untuk membantu Anda mengonfigurasi kebijakan otorisasi untuk komponen Anda.

Example Contoh kebijakan otorisasi

Contoh kebijakan otorisasi berikut memungkinkan komponen untuk menjeda dan melanjutkan semua komponen.

```
{
  "accessControl": {
    "aws.greengrass.ipc.lifecycle": {
      "com.example.MyLocalLifecycleComponent:lifecycle:1": {
        "policyDescription": "Allows access to pause/resume all components.",
        "operations": [
          "aws.greengrass#PauseComponent",
          "aws.greengrass#ResumeComponent"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
}
```

UpdateState

Memperbarui keadaan komponen pada perangkat inti.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`state`

Keadaan yang akan diatur. Enum ini, `LifecycleState`, memiliki nilai-nilai berikut:

- `RUNNING`
- `ERRORED`

Respons

Operasi ini tidak memberikan informasi apa pun dalam tanggapannya.

SubscribeToComponentUpdates

Berlangganlah untuk menerima notifikasi sebelum perangkat lunak inti AWS IoT Greengrass memperbarui komponen. Notifikasi tersebut menentukan apakah nukleus akan di-restart sebagai bagian dari pembaruan.

Inti tersebut mengirimkan notifikasi pembaruan hanya jika komponen kebijakan pembaruan komponen deployment menentukan untuk menotifikasi komponen. Perilaku defaultnya adalah menotifikasi komponen. Untuk informasi selengkapnya, lihat Buat deployment dan DeploymentComponentUpdatePolicyobjek yang dapat Anda berikan saat Anda memanggil CreateDeploymentoperasi.

> **Important**
>
> Deployment lokal tidak menotifikasi komponen sebelum pembaruan.

Operasi ini adalah operasi berlangganan di mana Anda berlangganan aliran pesan peristiwa. Untuk menggunakan operasi ini, tentukan bagian yang menangani respons aliran dengan fungsi yang menangani pesan peristiwa, kesalahan, dan penutupan aliran. Untuk informasi selengkapnya, lihat Berlangganan pengaliran peristiwa IPC.

Jenis pesan peristiwa: `ComponentUpdatePolicyEvents`

> **Tip**
>
> Anda dapat mengikuti tutorial untuk mempelajari cara mengembangkan komponen yang secara kondisional menunda pembaruan komponen. Untuk informasi selengkapnya, lihat Tutorial: Mengembangkan komponen Greengrass yang menunda pembaruan komponen.

Permintaan

Operasi ini tidak memiliki parameter apa pun.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`messages`

Aliran pesan notifikasi. Objek ini, `ComponentUpdatePolicyEvents`, berisi informasi berikut:

`preUpdateEvent`(Python:) `pre_update_event`

(Opsional) Sebuah peristiwa yang menunjukkan bahwa inti ingin memperbarui komponen. Anda dapat merespons dengan operasi DeferComponentUpdate untuk mengakui atau menunda pembaruan sampai komponen Anda siap untuk memulai ulang. Objek ini, `PreComponentUpdateEvent`, berisi informasi berikut:

`deploymentId`(Python:) `deployment_id`

ID deployment AWS IoT Greengrass yang memperbarui komponen.

`isGgcRestarting`(Python:) `is_ggc_restarting`

Apakah nukleus perlu me-restart untuk menerapkan pembaruan.

`postUpdateEvent`(Python:) `post_update_event`

(Opsional) Sebuah peristiwa yang menunjukkan bahwa inti telah memperbarui komponen. Objek ini, `PostComponentUpdateEvent`, berisi informasi berikut:

`deploymentId`(Python:) `deployment_id`

ID deployment AWS IoT Greengrass yang memperbarui komponen.

> **Note**
>
> Fitur ini membutuhkan v2.7.0 atau yang lebih baru dari komponen inti Greengrass.

DeferComponentUpdate

Mengakui atau menunda pembaruan komponen yang Anda temukan dengan SubscribeToComponentUpdates. Anda menentukan jumlah waktu untuk menunggu sebelum nukleus memeriksa lagi apakah komponen Anda siap untuk membiarkan pembaruan komponen dilanjutkan. Anda juga dapat menggunakan operasi ini untuk memberi tahu nukleus bahwa komponen Anda siap untuk pembaruan.

Jika komponen tidak menanggapi notifikasi pembaruan komponen, inti tersebut akan menunggu sejumlah waktu yang Anda tentukan dalam kebijakan pembaruan komponen deployment. Setelah batas waktu itu, nukleus akan melanjutkan deployment. Pembaruan komponen default adalah 60 detik. Untuk informasi selengkapnya, lihat Buat deployment dan DeploymentComponentUpdatePolicyobjek yang dapat Anda berikan saat Anda memanggil CreateDeploymentoperasi.

> **Tip**
>
> Anda dapat mengikuti tutorial untuk mempelajari cara mengembangkan komponen yang secara kondisional menunda pembaruan komponen. Untuk informasi selengkapnya, lihat Tutorial: Mengembangkan komponen Greengrass yang menunda pembaruan komponen.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`deploymentId`(Python:) `deployment_id`

ID deployment AWS IoT Greengrass yang akan ditunda.

`message`

(Opsional) Nama komponen untuk menunda pembaruan.

Default untuk nama komponen yang membuat permintaan.

`recheckAfterMs`(Python:) `recheck_after_ms`

Jumlah waktu dalam milidetik untuk menunda pembaruan. Nukleus menunggu selama jumlah waktu ini dan kemudian mengirimkan `PreComponentUpdateEvent` lainnya yang dapat Anda temukan dengan SubscribeToComponentUpdates.

Tentukan `0` untuk membenarkan pembaruan. Hal ini akan memberi tahu nukleus bahwa komponen Anda siap untuk diperbarui.

Default nol milidetik, yang berarti mengakui pembaruan.

Respons

Operasi ini tidak memberikan informasi apa pun dalam tanggapannya.

PauseComponent

Fitur ini tersedia untuk v2.4.0 dan yang lebih baru dari komponen inti Greengrass. AWS IoT Greengrasssaat ini tidak mendukung fitur ini di perangkat inti Windows.

Menjeda proses komponen pada perangkat inti. Untuk melanjutkan komponen, gunakan ResumeComponentoperasi.

Anda hanya dapat menjeda komponen generik. Jika Anda mencoba menjeda jenis komponen lainnya, operasi ini akan menampilkan file. `InvalidRequestError`

> **Note**
>
> Operasi ini tidak dapat menjeda proses kontainer, seperti kontainer Docker. Untuk menjeda dan melanjutkan container Docker, Anda dapat menggunakan perintah docker pause dan docker unpause.

Operasi ini tidak menjeda dependensi komponen atau komponen yang bergantung pada komponen yang Anda jeda. Pertimbangkan perilaku ini ketika Anda menjeda komponen yang merupakan ketergantungan komponen lain, karena komponen dependen mungkin mengalami masalah saat ketergantungannya dijeda.

Saat Anda memulai ulang atau mematikan komponen yang dijeda, seperti melalui penerapan, inti Greengrass melanjutkan komponen dan menjalankan siklus hidup shutdown-nya. Untuk informasi selengkapnya tentang memulai ulang komponen, lihat. RestartComponent

> **Important**
>
> Untuk menggunakan operasi ini, Anda harus menentukan kebijakan otorisasi yang memberikan izin untuk menggunakan operasi ini. Untuk informasi selengkapnya, lihat Otorisasi.

SDK (Versi Minimum)

Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk menjeda dan melanjutkan komponen.

SDK	Versi minimum	
AWS IoT Device SDKuntuk Java v2	v1.4.3	
AWS IoT Device SDKuntuk Python v2	v1.6.2	
AWS IoT Device SDKuntuk C++ v2	v1.13.1	
AWS IoT Device SDKuntuk JavaScript v2	v1.12.0	

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`componentName`(Python:) `component_name`

Nama komponen untuk jeda, yang harus menjadi komponen generik. Untuk informasi selengkapnya, lihat Jenis komponen.

Respons

Operasi ini tidak memberikan informasi apa pun dalam tanggapannya.

ResumeComponent

Fitur ini tersedia untuk v2.4.0 dan yang lebih baru dari komponen inti Greengrass. AWS IoT Greengrasssaat ini tidak mendukung fitur ini di perangkat inti Windows.

Melanjutkan proses komponen pada perangkat inti. Untuk menjeda komponen, gunakan PauseComponentoperasi.

Anda hanya dapat melanjutkan komponen yang dijeda. Jika Anda mencoba melanjutkan komponen yang tidak dijeda, operasi ini akan melempar file. `InvalidRequestError`

> **Important**
>
> Untuk menggunakan operasi ini, Anda harus menentukan kebijakan otorisasi yang memberikan izin untuk melakukannya. Untuk informasi selengkapnya, lihat Otorisasi.

SDK (Versi Minimum)

Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk menjeda dan melanjutkan komponen.

SDK	Versi minimum	
AWS IoT Device SDKuntuk Java v2	v1.4.3	
AWS IoT Device SDKuntuk Python v2	v1.6.2	
AWS IoT Device SDKuntuk C++ v2	v1.13.1	
AWS IoT Device SDKuntuk JavaScript v2	v1.12.0	

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`componentName`(Python:) `component_name`

Nama komponen yang akan dilanjutkan.

Respons

Operasi ini tidak memberikan informasi apa pun dalam tanggapannya.

Berinteraksilah dengan konfigurasi komponen

Layanan IPC konfigurasi komponen memungkinkan Anda melakukan hal berikut:

- Dapatkan dan atur parameter konfigurasi komponen.
- Berlangganan pembaruan konfigurasi komponen.
- Validasi pembaruan konfigurasi komponen sebelum nukleus menerapkannya.

Topik

SDK (Versi Minimum)

Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk berinteraksi dengan konfigurasi komponen.

SDK	Versi minimum	
AWS IoT Device SDKuntuk Java v2	v1.2.10	
AWS IoT Device SDKuntuk Python v2	v1.5.3	
AWS IoT Device SDKuntuk C++ v2	v1.17.0	
AWS IoT Device SDKuntuk JavaScript v2	v1.12.0	

GetConfiguration

Dapatkan nilai konfigurasi untuk komponen pada perangkat inti. Anda menentukan jalur kunci untuk mendapatkan nilai konfigurasi.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`componentName`(Python:) `component_name`

(Opsional) Nama komponen.

Default untuk nama komponen yang membuat permintaan.

`keyPath`(Python:) `key_path`

Path kunci untuk nilai konfigurasi. Tentukan daftar di mana setiap entri adalah kunci untuk satu tingkat dalam objek konfigurasi. Sebagai contoh, tentukan `["mqtt", "port"]` untuk mendapatkan nilai `port` dalam konfigurasi berikut.

```
{
  "mqtt": {
    "port": 443
  }
```

```
}
```

Untuk mendapatkan konfigurasi lengkap komponen, tentukan daftar kosong.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`componentName`(Python:) `component_name`

Nama komponen.

`value`

Konfigurasi yang diminta sebagai objek.

UpdateConfiguration

Memperbarui nilai konfigurasi untuk komponen ini pada perangkat inti.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`keyPath`(Python:) `key_path`

(Opsional) Path kunci untuk node kontainer (objek) yang akan diperbarui. Tentukan daftar di mana setiap entri adalah kunci untuk satu tingkat dalam objek konfigurasi. Sebagai contoh, tentukan path kunci `["mqtt"]` dan nilai merge `{ "port": 443 }` untuk mengatur nilai `port` dalam konfigurasi berikut.

```
{
  "mqtt": {
    "port": 443
  }
}
```

Path kunci harus menentukan node kontainer (objek) dalam konfigurasi tersebut. Jika node tidak ada dalam konfigurasi komponen, operasi ini akan membuatnya dan menetapkan nilainya ke objek tersebut di `valueToMerge`.

Default ke akar dari objek konfigurasi.

`timestamp`

Jangka waktu Unix saat ini dalam milidetik. Operasi ini menggunakan ini stempel waktu untuk menyelesaikan pembaruan bersamaan untuk kunci. Jika kunci dalam konfigurasi komponen memiliki cap waktu yang lebih besar dari cap waktu dalam permintaan, maka permintaan akan gagal.

`valueToMerge`(Python:) `value_to_merge`

Objek konfigurasi yang akan digabungkan di lokasi yang Anda tentukan di `keyPath`. Untuk informasi selengkapnya, lihat Perbarui konfigurasi komponen.

Respons

Operasi ini tidak memberikan informasi apa pun dalam tanggapannya.

SubscribeToConfigurationUpdate

Berlangganan untuk menerima notifikasi ketika konfigurasi komponen diperbarui. Ketika Anda berlangganan kunci, Anda akan menerima notifikasi ketika setiap anak dari kunci tersebut melakukan pembaruan.

Operasi ini adalah operasi berlangganan di mana Anda berlangganan aliran pesan peristiwa. Untuk menggunakan operasi ini, tentukan bagian yang menangani respons aliran dengan fungsi yang menangani pesan peristiwa, kesalahan, dan penutupan aliran. Untuk informasi selengkapnya, lihat Berlangganan pengaliran peristiwa IPC.

Jenis pesan peristiwa: `ConfigurationUpdateEvents`

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`componentName`(Python:) `component_name`

(Opsional) Nama komponen.

Default untuk nama komponen yang membuat permintaan.

`keyPath`(Python:) `key_path`

Path kunci untuk nilai konfigurasi yang akan dijadikan langganan. Tentukan daftar di mana setiap entri adalah kunci untuk satu tingkat dalam objek konfigurasi. Sebagai contoh, tentukan `["mqtt", "port"]` untuk mendapatkan nilai `port` dalam konfigurasi berikut.

```
{
  "mqtt": {
    "port": 443
  }
}
```

Untuk berlangganan pembaruan untuk semua nilai dalam konfigurasi komponen, tentukan daftar kosong.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`messages`

Aliran pesan notifikasi. Objek ini, `ConfigurationUpdateEvents`, berisi informasi berikut:

`configurationUpdateEvent`(Python:) `configuration_update_event`

Peristiwa pembaruan konfigurasi. Objek ini, `ConfigurationUpdateEvent`, berisi informasi berikut:

`componentName`(Python:) `component_name`

Nama komponen.

`keyPath`(Python:) `key_path`

Path kunci untuk nilai konfigurasi yang diperbarui.

SubscribeToValidateConfigurationUpdates

Berlangganlah untuk menerima notifikasi sebelum konfigurasi komponen diperbarui. Hal ini memungkinkan komponen memvalidasi pembaruan konfigurasinya sendiri. Gunakan operasi SendConfigurationValidityReport untuk memberitahu nukleus apakah konfigurasi tersebut valid atau tidak.

> **Important**
>
> Deployment lokal tidak menotifikasi komponen tentang pembaruan.

Operasi ini adalah operasi berlangganan di mana Anda berlangganan aliran pesan peristiwa. Untuk menggunakan operasi ini, tentukan bagian yang menangani respons aliran dengan fungsi yang menangani pesan peristiwa, kesalahan, dan penutupan aliran. Untuk informasi selengkapnya, lihat Berlangganan pengaliran peristiwa IPC.

Jenis pesan peristiwa: `ValidateConfigurationUpdateEvents`

Permintaan

Operasi ini tidak memiliki parameter apa pun.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`messages`

Aliran pesan notifikasi. Objek ini, `ValidateConfigurationUpdateEvents`, berisi informasi berikut:

`validateConfigurationUpdateEvent`(Python:) `validate_configuration_update_event`

Peristiwa pembaruan konfigurasi. Objek ini, `ValidateConfigurationUpdateEvent`, berisi informasi berikut:

`deploymentId`(Python:) `deployment_id`

ID deployment AWS IoT Greengrass yang memperbarui komponen.

`configuration`

Objek yang berisi konfigurasi baru.

SendConfigurationValidityReport

Beritahu nukleus apakah pembaruan konfigurasi komponen ini valid atau tidak. Deployment tersebut gagal jika Anda memberitahu nukleus bahwa konfigurasi baru tidak valid. Gunakan operasi

SubscribeToValidateConfigurationUpdates untuk berlangganan untuk memvalidasi pembaruan konfigurasi.

Jika komponen tidak menanggapi notifikasi pembaruan konfigurasi yang tervalidasi, inti akan menunggu sejumlah waktu yang Anda tentukan dalam kebijakan validasi konfigurasi deployment. Setelah batas waktu itu, nukleus akan melanjutkan deployment. Batas waktu validasi komponen default adalah 20 detik. Untuk informasi selengkapnya, lihat Buat deployment dan DeploymentConfigurationValidationPolicyobjek yang dapat Anda berikan saat Anda memanggil CreateDeploymentoperasi.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`configurationValidityReport`(Python:) `configuration_validity_report`

Laporan tersebut yang memberitahu inti apakah pembaruan konfigurasi valid atau tidak. Objek ini, `ConfigurationValidityReport`, berisi informasi berikut:

`status`

Status validitas. Enum ini, `ConfigurationValidityStatus`, memiliki nilai-nilai berikut:

- `ACCEPTED` – Konfigurasi ini valid dan nukleus dapat menerapkannya pada komponen ini.
- `REJECTED` – Konfigurasi ini tidak valid dan deployment ini gagal.

`deploymentId`(Python:) `deployment_id`

ID deployment AWS IoT Greengrass yang meminta pembaruan konfigurasi.

`message`

(Opsional) Pesan yang melaporkan mengapa konfigurasi tidak valid.

Respons

Operasi ini tidak memberikan informasi apa pun dalam tanggapannya.

Ambil nilai-nilai rahasia

Gunakan layanan IPC secret manager untuk mengambil nilai-nilai rahasia dari rahasia pada perangkat inti. Anda menggunakan komponen secret manager untuk men-deploy rahasia terenkripsi

ke perangkat inti. Kemudian, Anda dapat menggunakan operasi IPC untuk mendekripsi rahasia itu dan menggunakan nilainya dalam komponen kustom Anda.

Topik

SDK (Versi Minimum)

Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk mengambil nilai rahasia dari rahasia pada perangkat inti.

SDK	Versi minimum	
AWS IoT Device SDKuntuk Java v2	v1.2.10	
AWS IoT Device SDKuntuk Python v2	v1.5.3	
AWS IoT Device SDKuntuk C++ v2	v1.17.0	
AWS IoT Device SDKuntuk JavaScript v2	v1.12.0	

Otorisasi

Untuk menggunakan secrets manager dalam komponen kustom, Anda harus menentukan kebijakan otorisasi yang memungkinkan komponen Anda untuk mendapatkan nilai rahasia yang Anda simpan pada perangkat inti. Untuk informasi tentang cara menentukan kebijakan otorisasi, lihat Otorisasi komponen untuk melakukan operasi IPC.

Kebijakan otorisasi untuk secrets manager memiliki properti berikut.

Pengenal layanan IPC: `aws.greengrass.SecretManager`

Operasi	Deskripsi	Sumber daya
`aws.greengrass#GetSecretValue`	Memungkinkan komponen untuk mendapatkan nilai rahasia yang dienkripsi pada perangkat inti.	ARN rahasia Secrets Manager, atau * untuk memungkinkan akses ke semua rahasia.

Contoh kebijakan otorisasi

Anda dapat mereferensikan contoh kebijakan otorisasi berikut untuk membantu Anda mengonfigurasi kebijakan otorisasi untuk komponen Anda.

Example Contoh kebijakan otorisasi

Kebijakan otorisasi contoh berikut memungkinkan komponen untuk mendapatkan nilai rahasia apa pun pada perangkat inti.

> **Note**
>
> Kami merekomendasikan bahwa dalam lingkungan produksi, Anda mengurangi cakupan kebijakan otorisasi, sehingga komponen hanya mengambil rahasia yang digunakannya. Anda dapat mengubah wildcard * ke daftar ARN rahasia saat Anda men-deploy komponen.

```
{
  "accessControl": {
    "aws.greengrass.SecretManager": {
      "com.example.MySecretComponent:secrets:1": {
        "policyDescription": "Allows access to a secret.",
        "operations": [
          "aws.greengrass#GetSecretValue"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
```

```
}
```

GetSecretValue

Mendapatkan nilai rahasia yang Anda simpan pada perangkat inti.

Operasi ini mirip dengan operasi Secrets Manager yang dapat Anda gunakan untuk mendapatkan nilai rahasia di AWS Cloud. Untuk informasi selengkapnya, lihat GetSecretValue di dalam Referensi API AWS Secrets Manager.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`secretId`(Python:) `secret_id`

Nama layanan yang akan didapat. Anda dapat menentukan Amazon Resource Name (ARN) atau nama yang ramah untuk rahasia tersebut.

`versionId`(Python:) `version_id`

(Opsional) ID dari versi yang akan didapatkan.

Anda dapat menentukan `versionId` atau `versionStage`.

Jika Anda tidak menentukan `versionId` atau `versionStage`, operasi ini default ke versi dengan label `AWSCURRENT`.

`versionStage`(Python:) `version_stage`

(Opsional) Label penahapan dari versi yang akan didapatkan.

Anda dapat menentukan `versionId` atau `versionStage`.

Jika Anda tidak menentukan `versionId` atau `versionStage`, operasi ini default ke versi dengan label `AWSCURRENT`.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`secretId`(Python:) `secret_id`

ID dari rahasia tersebut.

`versionId`(Python:) `version_id`

ID dari versi rahasia ini.

`versionStage`(Python:) `version_stage`

Daftar label penahapan yang melekat pada versi rahasia ini.

`secretValue`(Python:) `secret_value`

Nilai dari versi rahasia ini. Objek ini, `SecretValue`, berisi informasi berikut.

`secretString`(Python:) `secret_string`

Bagian yang didekripsi dari informasi rahasia terlindungi yang Anda berikan kepada Secrets Manager sebagai string.

`secretBinary`(Python:) `secret_binary`

(Opsional) Bagian yang didekripsi dari informasi rahasia terlindungi yang Anda berikan kepada Secrets Manager sebagai data biner dalam bentuk himpunan byte. Properti ini berisi data biner sebagai string base64-encoded.

Properti ini tidak digunakan jika Anda membuat rahasia di konsol Secrets Manager.

Contoh-contoh

Contoh berikut ini mendemonstrasikan cara memanggil operasi ini dalam kode komponen kustom.

Java (IPC client V1)

Example Contoh: Dapatkan nilai rahasia

> **Note**
>
> Contoh ini menggunakan `IPCUtils` kelas untuk membuat koneksi ke layanan AWS IoT Greengrass Core IPC. Untuk informasi selengkapnya, lihat Connect ke layanan AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GetSecretValueResponseHandler;
```

```
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.model.GetSecretValueRequest;
import software.amazon.awssdk.aws.greengrass.model.GetSecretValueResponse;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class GetSecretValue {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        String secretArn = args[0];
        String versionStage = args[1];
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            GetSecretValueResponseHandler responseHandler =
                    GetSecretValue.getSecretValue(ipcClient, secretArn,
 versionStage);
            CompletableFuture<GetSecretValueResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                GetSecretValueResponse response =
 futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                response.getSecretValue().postFromJson();
                String secretString = response.getSecretValue().getSecretString();
                System.out.println("Successfully retrieved secret value: " +
 secretString);
            } catch (TimeoutException e) {
                System.err.println("Timeout occurred while retrieving secret: " +
 secretArn);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.println("Unauthorized error while retrieving secret:
 " + secretArn);
                } else {
                    throw e;
```

```
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static GetSecretValueResponseHandler
 getSecretValue(GreengrassCoreIPCClient greengrassCoreIPCClient, String secretArn,
 String versionStage) {
        GetSecretValueRequest getSecretValueRequest = new GetSecretValueRequest();
        getSecretValueRequest.setSecretId(secretArn);
        getSecretValueRequest.setVersionStage(versionStage);
        return greengrassCoreIPCClient.getSecretValue(getSecretValueRequest,
 Optional.empty());
    }
}
```

Python (IPC client V1)

Example Contoh: Dapatkan nilai rahasia

> **Note**
>
> Contoh ini mengasumsikan bahwa Anda menggunakan versi 1.5.4 atau yang lebih baru dari AWS IoT Device SDK untuk Python v2.

```
import json

import awsiot.greengrasscoreipc
from awsiot.greengrasscoreipc.model import (
    GetSecretValueRequest,
    GetSecretValueResponse,
    UnauthorizedError
)
```

```
secret_id = 'arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyGreengrassSecret-abcdef'
TIMEOUT = 10

ipc_client = awsiot.greengrasscoreipc.connect()

request = GetSecretValueRequest()
request.secret_id = secret_id
request.version_stage = 'AWSCURRENT'
operation = ipc_client.new_get_secret_value()
operation.activate(request)
future_response = operation.get_response()
response = future_response.result(TIMEOUT)
secret_json = json.loads(response.secret_value.secret_string)
# Handle secret value.
```

JavaScript

Example Contoh: Dapatkan nilai rahasia

```
import {
    GetSecretValueRequest,
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";

class GetSecretValue {
    private readonly secretId : string;
    private readonly versionStage : string;
    private ipcClient : greengrasscoreipc.Client

    constructor() {
        this.secretId = "<define_your_own_secretId>"
        this.versionStage = "<define_your_own_versionStage>"

        this.getSecretValue().then(r => console.log("Started workflow"));
    }

    private async getSecretValue() {
        try {
            this.ipcClient = await getIpcClient();

            const getSecretValueRequest : GetSecretValueRequest = {
```

```
                secretId: this.secretId,
                versionStage: this.versionStage,
            };

            const result = await
 this.ipcClient.getSecretValue(getSecretValueRequest);
            const secretString = result.secretValue.secretString;
            console.log("Successfully retrieved secret value: " + secretString)
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}

export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

const getSecretValue = new GetSecretValue();
```

Contoh-contoh

Gunakan contoh berikut untuk mempelajari cara menggunakan layanan IPC secret manager dalam komponen Anda.

Contoh: Cetak rahasia (Python, klien IPC V1)

Contoh komponen ini mencetak nilai suatu rahasia yang Anda deploy ke perangkat inti tersebut.

> **Important**
>
> Contoh komponen ini mencetak nilai rahasia, jadi gunakan hanya dengan rahasia yang menyimpan data uji. Jangan gunakan komponen ini untuk mencetak nilai rahasia yang menyimpan informasi penting.

Topik

Resep

Contoh resep berikut menentukan parameter konfigurasi ARN rahasia dan memungkinkan komponen untuk mendapatkan nilai rahasia apa pun pada perangkat inti.

> **Note**
>
> Kami merekomendasikan bahwa dalam lingkungan produksi, Anda mengurangi cakupan kebijakan otorisasi, sehingga komponen hanya mengambil rahasia yang digunakannya. Anda dapat mengubah wildcard * ke daftar ARN rahasia saat Anda men-deploy komponen.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PrintSecret",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Prints the value of an AWS Secrets Manager secret.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.SecretManager": {
      "VersionRequirement": "^2.0.0",
      "DependencyType": "HARD"
    }
  },
  "ComponentConfiguration": {
```

```
    "DefaultConfiguration": {
      "SecretArn": "",
      "accessControl": {
        "aws.greengrass.SecretManager": {
          "com.example.PrintSecret:secrets:1": {
            "policyDescription": "Allows access to a secret.",
            "operations": [
              "aws.greengrass#GetSecretValue"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk",
        "run": "python3 -u {artifacts:path}/print_secret.py \"{configuration:/
SecretArn}\""
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/print_secret.py \"{configuration:/
SecretArn}\""
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PrintSecret
ComponentVersion: 1.0.0
ComponentDescription: Prints the value of a Secrets Manager secret.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.SecretManager:
    VersionRequirement: "^2.0.0"
    DependencyType: HARD
ComponentConfiguration:
  DefaultConfiguration:
    SecretArn: ''
    accessControl:
      aws.greengrass.SecretManager:
        com.example.PrintSecret:secrets:1:
          policyDescription: Allows access to a secret.
          operations:
            - aws.greengrass#GetSecretValue
          resources:
            - "*"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: python3 -m pip install --user awsiotsdk
      run: python3 -u {artifacts:path}/print_secret.py "{configuration:/SecretArn}"
  - Platform:
      os: windows
    Lifecycle:
      install: py -3 -m pip install --user awsiotsdk
      run: py -3 -u {artifacts:path}/print_secret.py "{configuration:/SecretArn}"
```

Artifacts

Contoh aplikasi Python berikut menunjukkan cara menggunakan layanan IPC secret manager untuk mendapatkan nilai rahasia pada perangkat inti.

```
import concurrent.futures
import json
```

```
import sys
import traceback

import awsiot.greengrasscoreipc
from awsiot.greengrasscoreipc.model import (
    GetSecretValueRequest,
    GetSecretValueResponse,
    UnauthorizedError
)

TIMEOUT = 10

if len(sys.argv) == 1:
    print('Provide SecretArn in the component configuration.', file=sys.stdout)
    exit(1)

secret_id = sys.argv[1]

try:
    ipc_client = awsiot.greengrasscoreipc.connect()

    request = GetSecretValueRequest()
    request.secret_id = secret_id
    operation = ipc_client.new_get_secret_value()
    operation.activate(request)
    future_response = operation.get_response()

    try:
        response = future_response.result(TIMEOUT)
        secret_json = json.loads(response.secret_value.secret_string)
        print('Successfully got secret: ' + secret_id)
        print('Secret value: ' + str(secret_json))
    except concurrent.futures.TimeoutError:
        print('Timeout occurred while getting secret: ' + secret_id, file=sys.stderr)
    except UnauthorizedError as e:
        print('Unauthorized error while getting secret: ' + secret_id,
 file=sys.stderr)
        raise e
    except Exception as e:
        print('Exception while getting secret: ' + secret_id, file=sys.stderr)
        raise e
except Exception:
    print('Exception occurred when using IPC.', file=sys.stderr)
    traceback.print_exc()
```

```
    exit(1)
```

Penggunaan

Anda dapat menggunakan komponen contoh ini dengan komponen secret manager untuk men-deploy dan mencetak nilai rahasia pada perangkat inti Anda.

Untuk membuat, men-deploy, dan mencetak rahasia uji

1. Buat rahasia Secrets Manager dengan data uji.

   Linux or Unix

   ```
   aws secretsmanager create-secret \
     --name MyTestGreengrassSecret \
     --secret-string '{"my-secret-key": "my-secret-value"}'
   ```

   Windows Command Prompt (CMD)

   ```
   aws secretsmanager create-secret ^
     --name MyTestGreengrassSecret ^
     --secret-string '{"my-secret-key": "my-secret-value"}'
   ```

   PowerShell

   ```
   aws secretsmanager create-secret `
     --name MyTestGreengrassSecret `
     --secret-string '{"my-secret-key": "my-secret-value"}'
   ```

   Simpan ARN rahasia baru untuk digunakan dalam langkah-langkah berikut.

   Untuk informasi selengkapnya, lihat Membuat rahasia di Panduan Pengguna AWS Secrets Manager.

2. Deploy komponen secret manager (`aws.greengrass.SecretManager`) dengan pembaruan gabungan konfigurasi berikut. Tentukan ARN rahasia yang Anda buat sebelumnya.

   ```
   {
     "cloudSecrets": [
       {
   ```

```
      "arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestGreengrassSecret-abcdef"
    }
  ]
}
```

Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat atau perintah deployment Greengrass CLI.

3. Buat dan deploy komponen contoh pada bagian ini dengan pembaruan gabungan konfigurasi berikut. Tentukan ARN rahasia yang Anda buat sebelumnya.

```
{
  "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestGreengrassSecret",
  "accessControl": {
    "aws.greengrass.SecretManager": {
      "com.example.PrintSecret:secrets:1": {
        "policyDescription": "Allows access to a secret.",
        "operations": [
          "aws.greengrass#GetSecretValue"
        ],
        "resources": [
          "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestGreengrassSecret-abcdef"
        ]
      }
    }
  }
}
```

Lihat informasi yang lebih lengkap di Buat AWS IoT Greengrass komponen

4. Lihat log perangkat lunak inti AWS IoT Greengrass untuk memverifikasi bahwa deployment berhasil, dan melihat log komponen `com.example.PrintSecret` untuk mengetahui nilai rahasia yang dicetak. Lihat informasi yang lebih lengkap di Memantau AWS IoT Greengrass log.

Berinteraksi dengan bayangan lokal

Gunakan layanan IPC bayangan untuk berinteraksi dengan bayangan lokal pada perangkat. Perangkat yang Anda pilih untuk berinteraksi dapat berupa perangkat inti atau perangkat klien yang terhubung.

Untuk menggunakan operasi IPC ini, sertakan komponen shadow manager sebagai dependensi dalam komponen kustom Anda. Anda kemudian dapat menggunakan operasi IPC di komponen kustom Anda untuk berinteraksi dengan bayangan lokal di perangkat Anda melalui manajer bayangan. Untuk mengaktifkan komponen kustom untuk bereaksi terhadap perubahan di keadaan bayangan lokal, Anda juga dapat menggunakan layanan IPC publikasi/berlangganan untuk berlangganan bayangan peristiwa. Untuk informasi lebih lanjut tentang cara menggunakan layanan publikasi/berlangganan, lihat Pesan lokal publikasi/berlangganan.

> **Note**
>
> Untuk mengaktifkan perangkat inti berinteraksi dengan bayangan perangkat klien, Anda juga harus mengonfigurasi dan menerapkan komponen jembatan MQTT. Untuk informasi selengkapnya, lihat Mengaktifkan pengelola bayangan untuk berkomunikasi dengan perangkat klien.

Topik

SDK (Versi Minimum)

Tabel berikut menjelaskan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk berinteraksi dengan bayangan lokal.

SDK	Versi minimum	
AWS IoT Device SDKuntuk Java v2	v1.4.0	
AWS IoT Device SDKuntuk Python v2	v1.6.0	
AWS IoT Device SDKuntuk C++ v2	v1.17.0	
AWS IoT Device SDKuntuk JavaScript v2	v1.12.0	

Otorisasi

Untuk menggunakan layanan IPC bayangan dalam komponen kustom, Anda harus menentukan kebijakan otorisasi yang memungkinkan komponen Anda untuk berinteraksi dengan bayangan. Untuk informasi tentang cara menentukan kebijakan otorisasi, lihat Otorisasi komponen untuk melakukan operasi IPC.

Kebijakan otorisasi untuk interaksi bayangan memiliki properti berikut.

Pengenal layanan IPC: `aws.greengrass.ShadowManager`

Operasi	Deskripsi	Sumber daya
`aws.greengrass#GetThingShadow`	Memungkinkan komponen untuk mengambil bayangan suatu objek.	Salah satu string berikut: • `$aws/things/`*`thingName`*`/shadow/`, untuk memungkinkan akses ke bayangan perangkat klasik. • `$aws/things/`*`thingName`*`/shadow/n`

Operasi	Deskripsi	Sumber daya
		ame/ *shadowName* , untuk memungkinkan akses ke bayangan bernama. • *untuk memungkinkan akses ke semua bayangan.
`aws.greengrass#UpdateThingShadow`	Memungkinkan komponen untuk memperbarui bayangan suatu objek.	Salah satu string berikut: • `$aws/things/` *thingName* `/shadow/`, untuk memungkinkan akses ke bayangan perangkat klasik. • `$aws/things/` *thingName* `/shadow/name/` *shadowName* , untuk memungkinkan akses ke bayangan bernama. • *untuk memungkinkan akses ke semua bayangan.

Operasi	Deskripsi	Sumber daya
`aws.greengrass#DeleteThingShadow`	Memungkinkan komponen untuk menghapus bayangan suatu objek.	Salah satu string berikut: • `$aws/things/` *`thingName`* `/shadow/`, untuk mengizinkan akses ke bayangan perangkat klasik • `$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`* , untuk mengizinkan akses ke bayangan bernama • `*`, untuk memungkinkan akses ke semua bayangan.
`aws.greengrass#ListNamedShadowsForThing`	Memungkinkan komponen untuk mengambil daftar bayangan bernama untuk suatu objek.	Sebuah string nama objek yang memungkinkan akses ke objek untuk mencantumkan bayangannya. Gunakan `*` untuk memungkinkan akses ke semua hal.

Pengenal layanan IPC: `aws.greengrass.ipc.pubsub`

Operasi	Deskripsi	Sumber daya
`aws.greengrass#SubscribeToTopic`	Memungkinkan komponen untuk berlangganan pesan untuk topik yang Anda tentukan.	Salah satu string berikut: • *`shadowTopicPrefix`* `/get/accepted` • *`shadowTopicPrefix`* `/get/rejected`

Operasi	Deskripsi	Sumber daya
		• *shadowTopicPrefix* / delete/accepted • *shadowTopicPrefix* / delete/rejected • *shadowTopicPrefix* / update/accepted • *shadowTopicPrefix* / update/delta • *shadowTopicPrefix* / update/rejected Nilai awalan topik *shadowTopicPrefix* tergantung pada jenis bayangan: • Bayangan klasik: $aws/things/ *thingName* /shadow • Bayangan bernama: $aws/things/ *thingName* /shadow/name/ *shadowName* Gunakan * untuk mengizinkan akses semua topik. Di Greengrass nucleus v2.6.0 dan yang lebih baru, Anda dapat berlangganan topik yang berisi wildcard topik MQTT (dan). # + String topik ini mendukung wildcard topik MQTT sebagai karakter literal. Misalnya, jika

Operasi	Deskripsi	Sumber daya
		kebijakan otorisasi komponen memberikan akses ketest/topic/# , komponen dapat berlangganantest/topic/#, tetapi tidak dapat berlangganan. test/topic/filter

Variabel resep dalam kebijakan otorisasi bayangan lokal

Jika Anda menggunakan v2.6.0 atau yang lebih baru dari inti Greengrass, dan Anda menyetel opsi konfigurasi inti Greengrass ke, Anda dapat menggunakan variabel resep dalam kebijakan interpolateComponentConfigurationotorisasi. `true{iot:thingName}` Fitur ini memungkinkan Anda mengonfigurasi kebijakan otorisasi tunggal untuk sekelompok perangkat inti, di mana setiap perangkat inti hanya dapat mengakses bayangannya sendiri. Misalnya, Anda dapat mengizinkan akses komponen ke sumber daya berikut untuk operasi IPC bayangan.

```
$aws/things/{iot:thingName}/shadow/
```

Contoh kebijakan otorisasi

Anda dapat mereferensikan contoh kebijakan otorisasi berikut untuk membantu Anda mengonfigurasi kebijakan otorisasi untuk komponen Anda.

Example Contoh: Izinkan sekelompok perangkat inti berinteraksi dengan bayangan lokal

> **Important**
>
> Contoh ini menggunakan fitur yang tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass. Greengrass nucleus v2.6.0 menambahkan dukungan untuk sebagian besar variabel resep, seperti, dalam konfigurasi komponen. `{iot:thingName}` Untuk mengaktifkan fitur ini, atur opsi konfigurasi Greengrass nucleus ke interpolateComponentConfiguration. `true` Untuk contoh yang berfungsi untuk semua versi inti Greengrass, lihat contoh kebijakan otorisasi untuk perangkat inti tunggal.

Contoh kebijakan otorisasi berikut `com.example.MyShadowInteractionComponent` memungkinkan komponen berinteraksi dengan bayangan perangkat klasik dan bayangan bernama `myNamedShadow` untuk perangkat inti yang menjalankan komponen. Kebijakan ini juga memungkinkan komponen ini untuk menerima pesan pada topik lokal untuk bayangan ini.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ShadowManager": {
      "com.example.MyShadowInteractionComponent:shadow:1": {
        "policyDescription": "Allows access to shadows",
        "operations": [
          "aws.greengrass#GetThingShadow",
          "aws.greengrass#UpdateThingShadow",
          "aws.greengrass#DeleteThingShadow"
        ],
        "resources": [
          "$aws/things/{iot:thingName}/shadow",
          "$aws/things/{iot:thingName}/shadow/name/myNamedShadow"
        ]
      },
      "com.example.MyShadowInteractionComponent:shadow:2": {
        "policyDescription": "Allows access to things with shadows",
        "operations": [
          "aws.greengrass#ListNamedShadowsForThing"
        ],
        "resources": [
          "{iot:thingName}"
        ]
      }
    },
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyShadowInteractionComponent:pubsub:1": {
        "policyDescription": "Allows access to shadow pubsub topics",
        "operations": [
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "$aws/things/{iot:thingName}/shadow/get/accepted",
          "$aws/things/{iot:thingName}/shadow/name/myNamedShadow/get/accepted"
        ]
      }
```

```
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ShadowManager:
    'com.example.MyShadowInteractionComponent:shadow:1':
      policyDescription: 'Allows access to shadows'
      operations:
        - 'aws.greengrass#GetThingShadow'
        - 'aws.greengrass#UpdateThingShadow'
        - 'aws.greengrass#DeleteThingShadow'
      resources:
        - $aws/things/{iot:thingName}/shadow
        - $aws/things/{iot:thingName}/shadow/name/myNamedShadow
    'com.example.MyShadowInteractionComponent:shadow:2':
      policyDescription: 'Allows access to things with shadows'
      operations:
        - 'aws.greengrass#ListNamedShadowsForThing'
      resources:
        - '{iot:thingName}'
  aws.greengrass.ipc.pubsub:
    'com.example.MyShadowInteractionComponent:pubsub:1':
      policyDescription: 'Allows access to shadow pubsub topics'
      operations:
        - 'aws.greengrass#SubscribeToTopic'
      resources:
        - $aws/things/{iot:thingName}/shadow/get/accepted
        - $aws/things/{iot:thingName}/shadow/name/myNamedShadow/get/accepted
```

Example Contoh: Izinkan sekelompok perangkat inti berinteraksi dengan bayangan perangkat klien

> **Important**
>
> Fitur ini membutuhkan Greengrass nucleus v2.6.0 atau yang lebih baru, shadow manager v2.2.0 atau yang lebih baru, dan MQTT bridge v2.2.0 atau yang lebih baru. Anda harus mengkonfigurasi jembatan MQTT untuk mengaktifkan shadow manager untuk berkomunikasi dengan perangkat klien.

Contoh kebijakan otorisasi berikut memungkinkan komponen `com.example.MyShadowInteractionComponent` berinteraksi dengan semua bayangan perangkat untuk perangkat klien yang namanya dimulai`MyClientDevice`.

> **Note**
>
> Untuk mengaktifkan perangkat inti berinteraksi dengan bayangan perangkat klien, Anda juga harus mengonfigurasi dan menerapkan komponen jembatan MQTT. Untuk informasi selengkapnya, lihat Mengaktifkan pengelola bayangan untuk berkomunikasi dengan perangkat klien.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ShadowManager": {
      "com.example.MyShadowInteractionComponent:shadow:1": {
        "policyDescription": "Allows access to shadows",
        "operations": [
          "aws.greengrass#GetThingShadow",
          "aws.greengrass#UpdateThingShadow",
          "aws.greengrass#DeleteThingShadow"
        ],
        "resources": [
          "$aws/things/MyClientDevice*/shadow",
          "$aws/things/MyClientDevice*/shadow/name/*"
        ]
      },
      "com.example.MyShadowInteractionComponent:shadow:2": {
        "policyDescription": "Allows access to things with shadows",
        "operations": [
          "aws.greengrass#ListNamedShadowsForThing"
        ],
        "resources": [
          "MyClientDevice*"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ShadowManager:
    'com.example.MyShadowInteractionComponent:shadow:1':
      policyDescription: 'Allows access to shadows'
      operations:
        - 'aws.greengrass#GetThingShadow'
        - 'aws.greengrass#UpdateThingShadow'
        - 'aws.greengrass#DeleteThingShadow'
      resources:
        - $aws/things/MyClientDevice*/shadow
        - $aws/things/MyClientDevice*/shadow/name/*
    'com.example.MyShadowInteractionComponent:shadow:2':
      policyDescription: 'Allows access to things with shadows'
      operations:
        - 'aws.greengrass#ListNamedShadowsForThing'
      resources:
        - MyClientDevice*
```

Example Contoh: Izinkan perangkat inti tunggal berinteraksi dengan bayangan lokal

Kebijakan otorisasi contoh berikut memungkinkan komponen `com.example.MyShadowInteractionComponent` untuk berinteraksi dengan bayangan perangkat klasik dan bayangan bernama `myNamedShadow` untuk perangkat `MyThingName`. Kebijakan ini juga memungkinkan komponen ini untuk menerima pesan pada topik lokal untuk bayangan ini.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ShadowManager": {
      "com.example.MyShadowInteractionComponent:shadow:1": {
        "policyDescription": "Allows access to shadows",
        "operations": [
          "aws.greengrass#GetThingShadow",
          "aws.greengrass#UpdateThingShadow",
          "aws.greengrass#DeleteThingShadow"
        ],
        "resources": [
```

```
          "$aws/things/MyThingName/shadow",
          "$aws/things/MyThingName/shadow/name/myNamedShadow"
        ]
      },
      "com.example.MyShadowInteractionComponent:shadow:2": {
        "policyDescription": "Allows access to things with shadows",
        "operations": [
          "aws.greengrass#ListNamedShadowsForThing"
        ],
        "resources": [
          "MyThingName"
        ]
      }
    },
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyShadowInteractionComponent:pubsub:1": {
        "policyDescription": "Allows access to shadow pubsub topics",
        "operations": [
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "$aws/things/MyThingName/shadow/get/accepted",
          "$aws/things/MyThingName/shadow/name/myNamedShadow/get/accepted"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ShadowManager:
    'com.example.MyShadowInteractionComponent:shadow:1':
      policyDescription: 'Allows access to shadows'
      operations:
        - 'aws.greengrass#GetThingShadow'
        - 'aws.greengrass#UpdateThingShadow'
        - 'aws.greengrass#DeleteThingShadow'
      resources:
        - $aws/things/MyThingName/shadow
        - $aws/things/MyThingName/shadow/name/myNamedShadow
    'com.example.MyShadowInteractionComponent:shadow:2':
```

```
      policyDescription: 'Allows access to things with shadows'
      operations:
        - 'aws.greengrass#ListNamedShadowsForThing'
      resources:
        - MyThingName
  aws.greengrass.ipc.pubsub:
    'com.example.MyShadowInteractionComponent:pubsub:1':
      policyDescription: 'Allows access to shadow pubsub topics'
      operations:
        - 'aws.greengrass#SubscribeToTopic'
      resources:
        - $aws/things/MyThingName/shadow/get/accepted
        - $aws/things/MyThingName/shadow/name/myNamedShadow/get/accepted
```

Example Contoh: Izinkan sekelompok perangkat inti bereaksi terhadap perubahan status bayangan lokal

> **Important**
>
> Contoh ini menggunakan fitur yang tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass. Greengrass nucleus v2.6.0 menambahkan dukungan untuk sebagian besar variabel resep, seperti, dalam konfigurasi komponen. `{iot:thingName}` Untuk mengaktifkan fitur ini, atur opsi konfigurasi Greengrass nucleus ke interpolateComponentConfiguration. `true` Untuk contoh yang berfungsi untuk semua versi inti Greengrass, lihat contoh kebijakan otorisasi untuk perangkat inti tunggal.

Contoh kebijakan kontrol akses berikut memungkinkan kustom `com.example.MyShadowReactiveComponent` menerima pesan tentang `/update/delta` topik untuk bayangan perangkat klasik dan bayangan bernama `myNamedShadow` pada setiap perangkat inti yang menjalankan komponen.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyShadowReactiveComponent:pubsub:1": {
        "policyDescription": "Allows access to shadow pubsub topics",
```

```
        "operations": [
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "$aws/things/{iot:thingName}/shadow/update/delta",
          "$aws/things/{iot:thingName}/shadow/name/myNamedShadow/update/delta"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ipc.pubsub:
    "com.example.MyShadowReactiveComponent:pubsub:1":
      policyDescription: Allows access to shadow pubsub topics
      operations:
        - 'aws.greengrass#SubscribeToTopic'
      resources:
        - $aws/things/{iot:thingName}/shadow/update/delta
        - $aws/things/{iot:thingName}/shadow/name/myNamedShadow/update/delta
```

Example Contoh: Izinkan perangkat inti tunggal bereaksi terhadap perubahan status bayangan lokal

Contoh kebijakan kontrol akses berikut memungkinkan kustom `com.example.MyShadowReactiveComponent` menerima pesan tentang `/update/delta` topik untuk bayangan perangkat klasik dan bayangan bernama `myNamedShadow` untuk perangkat`MyThingName`.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyShadowReactiveComponent:pubsub:1": {
        "policyDescription": "Allows access to shadow pubsub topics",
        "operations": [
          "aws.greengrass#SubscribeToTopic"
        ],
```

```
        "resources": [
          "$aws/things/MyThingName/shadow/update/delta",
          "$aws/things/MyThingName/shadow/name/myNamedShadow/update/delta"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ipc.pubsub:
    "com.example.MyShadowReactiveComponent:pubsub:1":
      policyDescription: Allows access to shadow pubsub topics
      operations:
        - 'aws.greengrass#SubscribeToTopic'
      resources:
        - $aws/things/MyThingName/shadow/update/delta
        - $aws/things/MyThingName/shadow/name/myNamedShadow/update/delta
```

GetThingShadow

Dapatkan bayangan untuk objek tertentu.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`thingName`(Python:) `thing_name`

Nama sesuatu .

Tipe: `string`

`shadowName`(Python:) `shadow_name`

Nama bayangan. Untuk menentukan bayangan klasik benda itu, atur parameter ini ke string kosong ("").

> **Warning**
>
> AWS IoT GreengrassLayanan ini menggunakan bayangan `AWSManagedGreengrassV2Deployment` bernama untuk mengelola penerapan yang menargetkan perangkat inti individual. Bayangan bernama ini dicadangkan untuk digunakan oleh AWS IoT Greengrass layanan. Jangan perbarui atau hapus bayangan bernama ini.

Tipe: `string`

Respons

Tanggapan operasi ini memiliki informasi berikut:

`payload`

Dokumen keadaan respons sebagai gumpalan.

Jenis: `object` yang berisi informasi berikut:

`state`

Informasi keadaan.

Objek ini berisi informasi berikut.

`desired`

Properti keadaan dan nilai-nilai yang diminta untuk diperbarui di perangkat.

Jenis: `map` pasangan nilai kunci

`reported`

Properti keadaan dan nilai yang dilaporkan oleh perangkat.

Jenis: `map` pasangan nilai kunci

`delta`

Perbedaan antara properti dan nilai yang diinginkan dan yang dilaporkan. Properti ini hadir hanya jika keadaan `desired` dan `reported` berbeda.

Jenis: `map` pasangan nilai kunci

`metadata`

Cap waktu untuk setiap atribut dalam bagian `desired` dan `reported` agar Anda dapat menentukan kapan keadaan diperbarui.

Tipe: `string`

`timestamp`

Tanggal dan jangka waktu respons dihasilkan.

Tipe: `integer`

`clientToken`(Python:) `clientToken`

Token yang digunakan untuk mencocokkan permintaan tersebut dan respons yang sesuai.

Tipe: `string`

`version`

Versi dokumen bayangan lokal.

Tipe: `integer`

Kesalahan

Operasi ini dapat mengembalikan kesalahan berikut.

`InvalidArgumentsError`

Layanan bayangan lokal tidak dapat memvalidasi parameter permintaan. Hal ini dapat terjadi jika permintaan berisi JSON yang salah bentuk atau karakter yang tidak didukung.

`ResourceNotFoundError`

Dokumen bayangan lokal yang diminta tidak dapat ditemukan.

`ServiceError`

Terjadi kesalahan layanan internal, atau jumlah permintaan ke layanan IPC melebihi batas yang ditentukan dalam parameter konfigurasi `maxLocalRequestsPerSecondPerThing` dan `maxTotalLocalRequestsRate` di komponen manajer bayangan.

`UnauthorizedError`

Kebijakan otorisasi komponen tidak mencakup izin yang diperlukan untuk operasi ini.

Contoh-contoh

Contoh-contoh berikut ini menunjukkan cara memanggil operasi ini dalam kode komponen kustom.

Java (IPC client V1)

Example Contoh: Dapatkan bayangan benda

> **Note**
>
> Contoh ini menggunakan `IPCUtils` kelas untuk membuat koneksi ke layanan AWS IoT Greengrass Core IPC. Untuk informasi selengkapnya, lihat Connect ke layanan AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GetThingShadowResponseHandler;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.model.GetThingShadowRequest;
import software.amazon.awssdk.aws.greengrass.model.GetThingShadowResponse;
import software.amazon.awssdk.aws.greengrass.model.ResourceNotFoundError;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class GetThingShadow {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        // Use the current core device's name if thing name isn't set.
        String thingName = args[0].isEmpty() ? System.getenv("AWS_IOT_THING_NAME") :
 args[0];
        String shadowName = args[1];
```

```
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            GetThingShadowResponseHandler responseHandler =
                    GetThingShadow.getThingShadow(ipcClient, thingName,
 shadowName);
            CompletableFuture<GetThingShadowResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                GetThingShadowResponse response =
 futureResponse.get(TIMEOUT_SECONDS,
                        TimeUnit.SECONDS);
                String shadowPayload = new String(response.getPayload(),
 StandardCharsets.UTF_8);
                System.out.printf("Successfully got shadow %s/%s: %s%n", thingName,
 shadowName,
                        shadowPayload);
            } catch (TimeoutException e) {
                System.err.printf("Timeout occurred while getting shadow: %s/%s%n",
 thingName,
                        shadowName);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.printf("Unauthorized error while getting shadow: %s/
%s%n",
                            thingName, shadowName);
                } else if (e.getCause() instanceof ResourceNotFoundError) {
                    System.err.printf("Unable to find shadow to get: %s/%s%n",
 thingName,
                            shadowName);
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }
```

```
    public static GetThingShadowResponseHandler
 getThingShadow(GreengrassCoreIPCClient greengrassCoreIPCClient, String thingName,
 String shadowName) {
        GetThingShadowRequest getThingShadowRequest = new GetThingShadowRequest();
        getThingShadowRequest.setThingName(thingName);
        getThingShadowRequest.setShadowName(shadowName);
        return greengrassCoreIPCClient.getThingShadow(getThingShadowRequest,
 Optional.empty());
    }
}
```

Python (IPC client V1)

Example Contoh: Dapatkan bayangan benda

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import GetThingShadowRequest

TIMEOUT = 10

def sample_get_thing_shadow_request(thingName, shadowName):
    try:
        # set up IPC client to connect to the IPC server
        ipc_client = awsiot.greengrasscoreipc.connect()

        # create the GetThingShadow request
        get_thing_shadow_request = GetThingShadowRequest()
        get_thing_shadow_request.thing_name = thingName
        get_thing_shadow_request.shadow_name = shadowName

        # retrieve the GetThingShadow response after sending the request to the IPC
 server
        op = ipc_client.new_get_thing_shadow()
        op.activate(get_thing_shadow_request)
        fut = op.get_response()

        result = fut.result(TIMEOUT)
        return result.payload

    except InvalidArgumentsError as e:
        # add error handling
        ...
```

```
    # except ResourceNotFoundError | UnauthorizedError | ServiceError
```

JavaScript

Example Contoh: Dapatkan bayangan benda

```
import {
    GetThingShadowRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';

class GetThingShadow {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private shadowName: string;

    constructor() {
        // Define args parameters here
        this.thingName = "<define_your_own_thingName>";
        this.shadowName = "<define_your_own_shadowName>";
        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }

        try {
            await this.handleGetThingShadowOperation(this.thingName,
                this.shadowName);
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleGetThingShadowOperation(
        thingName: string,
        shadowName: string
    ) {
```

```
        const request: GetThingShadowRequest = {
            thingName: thingName,
            shadowName: shadowName
        };
        const response = await this.ipcClient.getThingShadow(request);
    }
}

export async function getIpcClient() {
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use caseså
        throw err
    }
}

const startScript = new GetThingShadow();
```

UpdateThingShadow

Perbarui bayangan untuk objek tertentu. Jika bayangan tidak ada, satu dibuat.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`thingName`(Python:) `thing_name`

Nama sesuatu .

Tipe: `string`

`shadowName`(Python:) `shadow_name`

Nama bayangan. Untuk menentukan bayangan klasik benda itu, atur parameter ini ke string kosong ("").

> **Warning**
>
> AWS IoT GreengrassLayanan ini menggunakan bayangan `AWSManagedGreengrassV2Deployment` bernama untuk mengelola penerapan yang menargetkan perangkat inti individual. Bayangan bernama ini dicadangkan untuk digunakan oleh AWS IoT Greengrass layanan. Jangan perbarui atau hapus bayangan bernama ini.

Tipe: `string`

`payload`

Dokumen keadaan permintaan sebagai gumpalan.

Jenis: `object` yang berisi informasi berikut:

`state`

Informasi keadaan yang akan diperbarui. Operasi IPC ini hanya mempengaruhi kolom tertentu.

Objek ini berisi informasi berikut. Biasanya, Anda akan menggunakan properti `desired` atau properti `reported`, tetapi tidak keduanya dalam permintaan yang sama.

`desired`

Properti keadaan dan nilai-nilai yang diminta untuk diperbarui di perangkat.

Jenis: `map` pasangan nilai kunci

`reported`

Properti keadaan dan nilai yang dilaporkan oleh perangkat.

Jenis: `map` pasangan nilai kunci

`clientToken`(Python:) `client_token`

(Opsional) Token yang digunakan untuk mencocokkan permintaan dan respons yang sesuai dengan token klien.

Tipe: `string`

`version`

(Opsional) Versi dokumen bayangan lokal yang akan diperbarui. Layanan bayangan akan memproses pembaruan hanya jika versi tertentu cocok dengan versi terbaru yang dimilikinya.

Tipe: `integer`

Respons

Tanggapan operasi ini memiliki informasi berikut:

`payload`

Dokumen keadaan respons sebagai gumpalan.

Jenis: `object` yang berisi informasi berikut:

`state`

Informasi keadaan.

Objek ini berisi informasi berikut.

`desired`

Properti keadaan dan nilai-nilai yang diminta untuk diperbarui di perangkat.

Jenis: `map` pasangan nilai kunci

`reported`

Properti keadaan dan nilai yang dilaporkan oleh perangkat.

Jenis: `map` pasangan nilai kunci

`delta`

Properti keadaan dan nilai yang dilaporkan oleh perangkat.

Jenis: `map` pasangan nilai kunci

`metadata`

Cap waktu untuk setiap atribut dalam bagian `desired` dan `reported` agar Anda dapat menentukan kapan keadaan diperbarui.

Tipe: `string`

`timestamp`

Tanggal dan jangka waktu respons dihasilkan.

Tipe: `integer`

`clientToken`(Python:) `client_token`

Token yang digunakan untuk mencocokkan permintaan dan respons yang sesuai.

Tipe: `string`

`version`

Versi dokumen bayangan lokal setelah pembaruan selesai.

Tipe: `integer`

Kesalahan

Operasi ini dapat mengembalikan kesalahan berikut.

`ConflictError`

Layanan bayangan lokal mengalami konflik versi selama operasi pembaruan. Hal ini terjadi ketika versi dalam permintaan muatan tidak cocok dengan versi dalam dokumen bayangan lokal terbaru yang tersedia.

`InvalidArgumentsError`

Layanan bayangan lokal tidak dapat memvalidasi parameter permintaan. Hal ini dapat terjadi jika permintaan berisi JSON yang salah bentuk atau karakter yang tidak didukung.

`payload` yang valid memiliki properti berikut:

- Simpul `state` ada, dan merupakan objek yang berisi informasi keadaan `desired` atau `reported`.
- Node `desired` dan `reported` merupakan objek atau tidak sah. Setidaknya salah satu dari objek-objek ini harus berisi informasi keadaan yang valid.
- Kedalaman objek `desired` dan `reported` tidak dapat melebihi delapan node.
- Panjang nilai `clientToken` tidak dapat melebihi 64 karakter.
- Nilai `version` harus 1 atau lebih tinggi.

`ServiceError`

Terjadi kesalahan layanan internal, atau jumlah permintaan ke layanan IPC melebihi batas yang ditentukan dalam parameter konfigurasi `maxLocalRequestsPerSecondPerThing` dan `maxTotalLocalRequestsRate` di komponen manajer bayangan.

`UnauthorizedError`

Kebijakan otorisasi komponen tidak mencakup izin yang diperlukan untuk operasi ini.

Contoh-contoh

Contoh-contoh berikut ini menunjukkan cara memanggil operasi ini dalam kode komponen kustom.

Java (IPC client V1)

Example Contoh: Perbarui bayangan sesuatu

> **Note**
>
> Contoh ini menggunakan `IPCUtils` kelas untuk membuat koneksi ke layanan AWS IoT Greengrass Core IPC. Untuk informasi selengkapnya, lihat Connect ke layanan AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.UpdateThingShadowResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.aws.greengrass.model.UpdateThingShadowRequest;
import software.amazon.awssdk.aws.greengrass.model.UpdateThingShadowResponse;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;
```

```
public class UpdateThingShadow {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        // Use the current core device's name if thing name isn't set.
        String thingName = args[0].isEmpty() ? System.getenv("AWS_IOT_THING_NAME") :
 args[0];
        String shadowName = args[1];
        byte[] shadowPayload = args[2].getBytes(StandardCharsets.UTF_8);
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            UpdateThingShadowResponseHandler responseHandler =
                    UpdateThingShadow.updateThingShadow(ipcClient, thingName,
 shadowName,
                            shadowPayload);
            CompletableFuture<UpdateThingShadowResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                System.out.printf("Successfully updated shadow: %s/%s%n", thingName,
 shadowName);
            } catch (TimeoutException e) {
                System.err.printf("Timeout occurred while updating shadow: %s/%s%n",
 thingName,
                        shadowName);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.printf("Unauthorized error while updating shadow: %s/
%s%n",
                            thingName, shadowName);
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
```

```
    }

    public static UpdateThingShadowResponseHandler
 updateThingShadow(GreengrassCoreIPCClient greengrassCoreIPCClient, String
 thingName, String shadowName, byte[] shadowPayload) {
        UpdateThingShadowRequest updateThingShadowRequest = new
 UpdateThingShadowRequest();
        updateThingShadowRequest.setThingName(thingName);
        updateThingShadowRequest.setShadowName(shadowName);
        updateThingShadowRequest.setPayload(shadowPayload);
        return greengrassCoreIPCClient.updateThingShadow(updateThingShadowRequest,
                Optional.empty());
    }
}
```

Python (IPC client V1)

Example Contoh: Perbarui bayangan sesuatu

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import UpdateThingShadowRequest

TIMEOUT = 10

def sample_update_thing_shadow_request(thingName, shadowName, payload):
    try:
        # set up IPC client to connect to the IPC server
        ipc_client = awsiot.greengrasscoreipc.connect()

        # create the UpdateThingShadow request
        update_thing_shadow_request = UpdateThingShadowRequest()
        update_thing_shadow_request.thing_name = thingName
        update_thing_shadow_request.shadow_name = shadowName
        update_thing_shadow_request.payload = payload

        # retrieve the UpdateThingShadow response after sending the request to the
 IPC server
        op = ipc_client.new_update_thing_shadow()
        op.activate(update_thing_shadow_request)
        fut = op.get_response()

        result = fut.result(TIMEOUT)
        return result.payload
```

```
    except InvalidArgumentsError as e:
        # add error handling
    ...
    # except ConflictError | UnauthorizedError | ServiceError
```

JavaScript

Example Contoh: Perbarui bayangan sesuatu

```
import {
    UpdateThingShadowRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';

class UpdateThingShadow {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private shadowName: string;
    private shadowDocumentStr: string;

    constructor() {
        // Define args parameters here

        this.thingName = "<define_your_own_thingName>";
        this.shadowName = "<define_your_own_shadowName>";
        this.shadowDocumentStr = "<define_your_own_payload>";

        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }

        try {
            await this.handleUpdateThingShadowOperation(
                this.thingName,
                this.shadowName,
```

```
                this.shadowDocumentStr);
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleUpdateThingShadowOperation(
        thingName: string,
        shadowName: string,
        payloadStr: string
    ) {
        const request: UpdateThingShadowRequest = {
            thingName: thingName,
            shadowName: shadowName,
            payload: payloadStr
        }
        // make the UpdateThingShadow request
        const response = await this.ipcClient.updateThingShadow(request);
    }
}

export async function getIpcClient() {
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

const startScript = new UpdateThingShadow();
```

DeleteThingShadow

Menghapus bayangan untuk objek yang ditentukan.

Dimulai di shadow manager v2.0.4, menghapus bayangan menambah nomor versi. Misalnya, ketika Anda menghapus bayangan `MyThingShadow` di versi 1, versi bayangan yang dihapus adalah 2. Jika Anda kemudian membuat ulang bayangan dengan nama`MyThingShadow`, versi untuk bayangan itu adalah 3.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`thingName`(Python:) `thing_name`

Nama sesuatu .

Tipe: `string`

`shadowName`(Python:) `shadow_name`

Nama bayangan. Untuk menentukan bayangan klasik benda itu, atur parameter ini ke string kosong (`""`).

> **Warning**
>
> AWS IoT GreengrassLayanan ini menggunakan bayangan `AWSManagedGreengrassV2Deployment` bernama untuk mengelola penerapan yang menargetkan perangkat inti individual. Bayangan bernama ini dicadangkan untuk digunakan oleh AWS IoT Greengrass layanan. Jangan perbarui atau hapus bayangan bernama ini.

Tipe: `string`

Respons

Tanggapan operasi ini memiliki informasi berikut:

`payload`

Dokumen keadaan tanpa respons.

Kesalahan

Operasi ini dapat mengembalikan kesalahan berikut.

`InvalidArgumentsError`

Layanan bayangan lokal tidak dapat memvalidasi parameter permintaan. Hal ini dapat terjadi jika permintaan berisi JSON yang salah bentuk atau karakter yang tidak didukung.

`ResourceNotFoundError`

Dokumen bayangan lokal yang diminta tidak dapat ditemukan.

`ServiceError`

Terjadi kesalahan layanan internal, atau jumlah permintaan ke layanan IPC melebihi batas yang ditentukan dalam parameter konfigurasi `maxLocalRequestsPerSecondPerThing` dan `maxTotalLocalRequestsRate` di komponen manajer bayangan.

`UnauthorizedError`

Kebijakan otorisasi komponen tidak mencakup izin yang diperlukan untuk operasi ini.

Contoh-contoh

Contoh-contoh berikut ini menunjukkan cara memanggil operasi ini dalam kode komponen kustom.

Java (IPC client V1)

Example Contoh: Hapus bayangan benda

> **Note**
>
> Contoh ini menggunakan `IPCUtils` kelas untuk membuat koneksi ke layanan AWS IoT Greengrass Core IPC. Untuk informasi selengkapnya, lihat Connect ke layanan AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.DeleteThingShadowResponseHandler;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
```

```
import software.amazon.awssdk.aws.greengrass.model.DeleteThingShadowRequest;
import software.amazon.awssdk.aws.greengrass.model.DeleteThingShadowResponse;
import software.amazon.awssdk.aws.greengrass.model.ResourceNotFoundError;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class DeleteThingShadow {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        // Use the current core device's name if thing name isn't set.
        String thingName = args[0].isEmpty() ? System.getenv("AWS_IOT_THING_NAME") :
 args[0];
        String shadowName = args[1];
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            DeleteThingShadowResponseHandler responseHandler =
                    DeleteThingShadow.deleteThingShadow(ipcClient, thingName,
 shadowName);
            CompletableFuture<DeleteThingShadowResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                System.out.printf("Successfully deleted shadow: %s/%s%n", thingName,
 shadowName);
            } catch (TimeoutException e) {
                System.err.printf("Timeout occurred while deleting shadow: %s/%s%n",
 thingName,
                        shadowName);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.printf("Unauthorized error while deleting shadow: %s/
%s%n",
                            thingName, shadowName);
                } else if (e.getCause() instanceof ResourceNotFoundError) {
```

```
                    System.err.printf("Unable to find shadow to delete: %s/%s%n",
 thingName,
                            shadowName);
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static DeleteThingShadowResponseHandler
 deleteThingShadow(GreengrassCoreIPCClient greengrassCoreIPCClient, String
 thingName, String shadowName) {
        DeleteThingShadowRequest deleteThingShadowRequest = new
 DeleteThingShadowRequest();
        deleteThingShadowRequest.setThingName(thingName);
        deleteThingShadowRequest.setShadowName(shadowName);
        return greengrassCoreIPCClient.deleteThingShadow(deleteThingShadowRequest,
                Optional.empty());
    }
}
```

Python (IPC client V1)

Example Contoh: Hapus bayangan benda

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import DeleteThingShadowRequest

TIMEOUT = 10

def sample_delete_thing_shadow_request(thingName, shadowName):
    try:
        # set up IPC client to connect to the IPC server
        ipc_client = awsiot.greengrasscoreipc.connect()

        # create the DeleteThingShadow request
```

```
        delete_thing_shadow_request = DeleteThingShadowRequest()
        delete_thing_shadow_request.thing_name = thingName
        delete_thing_shadow_request.shadow_name = shadowName

        # retrieve the DeleteThingShadow response after sending the request to the
 IPC server
        op = ipc_client.new_delete_thing_shadow()
        op.activate(delete_thing_shadow_request)
        fut = op.get_response()

        result = fut.result(TIMEOUT)
        return result.payload

    except InvalidArgumentsError as e:
        # add error handling
    ...
    # except ResourceNotFoundError | UnauthorizedError | ServiceError
```

JavaScript

Example Contoh: Hapus bayangan benda

```
import {
    DeleteThingShadowRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';

class DeleteThingShadow {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private shadowName: string;

    constructor() {
        // Define args parameters here
        this.thingName = "<define_your_own_thingName>";
        this.shadowName = "<define_your_own_shadowName>";
        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
```

```
            throw err
        }

        try {
            await this.handleDeleteThingShadowOperation(this.thingName,
 this.shadowName)
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleDeleteThingShadowOperation(thingName: string, shadowName: string) {
        const request: DeleteThingShadowRequest = {
            thingName: thingName,
            shadowName: shadowName
        }
        // make the DeleteThingShadow request
        const response = await this.ipcClient.deleteThingShadow(request);
    }
}

export async function getIpcClient() {
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

const startScript = new DeleteThingShadow();
```

ListNamedShadowsForThing

Daftar bayangan bernama untuk objek yang ditentukan.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`thingName`(Python:) `thing_name`

Nama sesuatu .

Tipe: `string`

`pageSize`(Python:) `page_size`

(Opsional) Jumlah nama bayangan yag akan dikembalikan pada setiap panggilan.

Tipe: `integer`

Default: 25

Maksimum: 100

`nextToken`(Python:) `next_token`

(Opsional) Token untuk mengambil rangkaian hasil berikutnya. Nilai ini dikembalikan pada hasil berhalaman dan digunakan dalam panggilan yang mengembalikan halaman berikutnya.

Tipe: `string`

Respons

Tanggapan operasi ini memiliki informasi berikut:

`results`

Daftar nama bayangan.

Tipe: `array`

`timestamp`

(Opsional) Tanggal dan waktu respons tersebut dihasilkan.

Tipe: `integer`

`nextToken`(Python:) `next_token`

(Opsional) Nilai token yang akan digunakan dalam permintaan berhalaman untuk mengambil halaman berikutnya secara berurutan. Token ini tidak hadir ketika tidak terdapat lagi nama bayangan yang akan dikembalikan.

Tipe: `string`

> **Note**
>
> Jika ukuran halaman yang diminta persis sesuai dengan jumlah nama bayangan di respons, maka token ini hadir; Namun, ketika digunakan, token tersebut akan mengembalikan daftar kosong.

Kesalahan

Operasi ini dapat mengembalikan kesalahan berikut.

`InvalidArgumentsError`

Layanan bayangan lokal tidak dapat memvalidasi parameter permintaan. Hal ini dapat terjadi jika permintaan berisi JSON yang salah bentuk atau karakter yang tidak didukung.

`ResourceNotFoundError`

Dokumen bayangan lokal yang diminta tidak dapat ditemukan.

`ServiceError`

Terjadi kesalahan layanan internal, atau jumlah permintaan ke layanan IPC melebihi batas yang ditentukan dalam parameter konfigurasi `maxLocalRequestsPerSecondPerThing` dan `maxTotalLocalRequestsRate` di komponen manajer bayangan.

`UnauthorizedError`

Kebijakan otorisasi komponen tidak mencakup izin yang diperlukan untuk operasi ini.

Contoh-contoh

Contoh-contoh berikut ini menunjukkan cara memanggil operasi ini dalam kode komponen kustom.

Java (IPC client V1)

Example Contoh: Daftar benda bernama bayangan

> **Note**
>
> Contoh ini menggunakan `IPCUtils` kelas untuk membuat koneksi ke layanan AWS IoT Greengrass Core IPC. Untuk informasi selengkapnya, lihat Connect ke layanan AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import
 software.amazon.awssdk.aws.greengrass.ListNamedShadowsForThingResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.ListNamedShadowsForThingRequest;
import
 software.amazon.awssdk.aws.greengrass.model.ListNamedShadowsForThingResponse;
import software.amazon.awssdk.aws.greengrass.model.ResourceNotFoundError;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class ListNamedShadowsForThing {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        // Use the current core device's name if thing name isn't set.
        String thingName = args[0].isEmpty() ? System.getenv("AWS_IOT_THING_NAME") :
 args[0];
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
```

```
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            List<String> namedShadows = new ArrayList<>();
            String nextToken = null;
            try {
                // Send additional requests until there's no pagination token in the
 response.
                do {
                    ListNamedShadowsForThingResponseHandler responseHandler =

 ListNamedShadowsForThing.listNamedShadowsForThing(ipcClient, thingName,
                                    nextToken, 25);
                    CompletableFuture<ListNamedShadowsForThingResponse>
 futureResponse =
                            responseHandler.getResponse();
                    ListNamedShadowsForThingResponse response =
                            futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                    List<String> responseNamedShadows = response.getResults();
                    namedShadows.addAll(responseNamedShadows);
                    nextToken = response.getNextToken();
                } while (nextToken != null);
                System.out.printf("Successfully got named shadows for thing %s: %s
%n", thingName,
                        String.join(",", namedShadows));
            } catch (TimeoutException e) {
                System.err.println("Timeout occurred while listing named shadows for
 thing: " + thingName);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.println("Unauthorized error while listing named
 shadows for " +
                            "thing: " + thingName);
                } else if (e.getCause() instanceof ResourceNotFoundError) {
                    System.err.println("Unable to find thing to list named shadows:
 " + thingName);
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
```

```
            System.exit(1);
        }
    }

    public static ListNamedShadowsForThingResponseHandler
 listNamedShadowsForThing(GreengrassCoreIPCClient greengrassCoreIPCClient, String
 thingName, String nextToken, int pageSize) {
        ListNamedShadowsForThingRequest listNamedShadowsForThingRequest =
                new ListNamedShadowsForThingRequest();
        listNamedShadowsForThingRequest.setThingName(thingName);
        listNamedShadowsForThingRequest.setNextToken(nextToken);
        listNamedShadowsForThingRequest.setPageSize(pageSize);
        return
 greengrassCoreIPCClient.listNamedShadowsForThing(listNamedShadowsForThingRequest,
                Optional.empty());
    }
}
```

Python (IPC client V1)

Example Contoh: Daftar benda bernama bayangan

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import ListNamedShadowsForThingRequest

TIMEOUT = 10

def sample_list_named_shadows_for_thing_request(thingName, nextToken, pageSize):
    try:
        # set up IPC client to connect to the IPC server
        ipc_client = awsiot.greengrasscoreipc.connect()

        # create the ListNamedShadowsForThingRequest request
        list_named_shadows_for_thing_request = ListNamedShadowsForThingRequest()
        list_named_shadows_for_thing_request.thing_name = thingName
        list_named_shadows_for_thing_request.next_token = nextToken
        list_named_shadows_for_thing_request.page_size = pageSize

        # retrieve the ListNamedShadowsForThingRequest response after sending the
 request to the IPC server
        op = ipc_client.new_list_named_shadows_for_thing()
        op.activate(list_named_shadows_for_thing_request)
        fut = op.get_response()
```

```
        list_result = fut.result(TIMEOUT)

        # additional returned fields
        timestamp = list_result.timestamp
        next_token = result.next_token
        named_shadow_list = list_result.results

        return named_shadow_list, next_token, timestamp

    except InvalidArgumentsError as e:
        # add error handling
    ...
    # except ResourceNotFoundError | UnauthorizedError | ServiceError
```

JavaScript

Example Contoh: Daftar benda bernama bayangan

```
import {
    ListNamedShadowsForThingRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';

class listNamedShadowsForThing {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private pageSizeStr: string;
    private nextToken: string;

    constructor() {
        // Define args parameters here
        this.thingName = "<define_your_own_thingName>";
        this.pageSizeStr = "<define_your_own_pageSize>";
        this.nextToken = "<define_your_own_token>";
        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
            throw err
```

```
        }

        try {
            await this.handleListNamedShadowsForThingOperation(this.thingName,
                this.nextToken, this.pageSizeStr);
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleListNamedShadowsForThingOperation(
        thingName: string,
        nextToken: string,
        pageSizeStr: string
    ) {
        let request: ListNamedShadowsForThingRequest = {
            thingName: thingName,
            nextToken: nextToken,
        };
        if (pageSizeStr) {
            request.pageSize = parseInt(pageSizeStr);
        }
        // make the ListNamedShadowsForThing request
        const response = await this.ipcClient.listNamedShadowsForThing(request);
        const shadowNames = response.results;
    }
}

export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}
```

```
const startScript = new listNamedShadowsForThing();
```

Kelola penerapan dan komponen lokal

> **Note**
>
> Fitur ini tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass.

Gunakan layanan Greengrass CLI IPC untuk mengelola penerapan lokal dan komponen Greengrass pada perangkat inti.

Untuk menggunakan operasi IPC ini, sertakan versi 2.6.0 atau yang lebih baru dari komponen CLI Greengrass sebagai ketergantungan dalam komponen kustom Anda. Anda kemudian dapat menggunakan operasi IPC di komponen kustom Anda untuk melakukan hal berikut:

- Buat penerapan lokal untuk memodifikasi dan mengonfigurasi komponen Greengrass pada perangkat inti.
- Mulai ulang dan hentikan komponen Greengrass pada perangkat inti.
- Buat kata sandi yang dapat Anda gunakan untuk masuk ke konsol debug lokal.

Topik

- SDK (Versi Minimum)
- Otorisasi
- CreateLocalDeployment
- ListLocalDeployments
- GetLocalDeploymentStatus
- ListComponents
- GetComponentDetails
- RestartComponent
- StopComponent
- CreateDebugPassword

SDK (Versi Minimum)

Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk berinteraksi dengan layanan Greengrass CLI IPC.

SDK	Versi minimum	
AWS IoT Device SDKuntuk Java v2	v1.2.10	
AWS IoT Device SDKuntuk Python v2	v1.5.3	
AWS IoT Device SDKuntuk C++ v2	v1.17.0	
AWS IoT Device SDKuntuk JavaScript v2	v1.12.0	

Otorisasi

Untuk menggunakan layanan Greengrass CLI IPC dalam komponen kustom, Anda harus menentukan kebijakan otorisasi yang memungkinkan komponen Anda mengelola penerapan dan komponen lokal. Untuk informasi tentang cara menentukan kebijakan otorisasi, lihat Otorisasi komponen untuk melakukan operasi IPC.

Kebijakan otorisasi untuk CLI Greengrass memiliki properti berikut.

Pengenal layanan IPC: `aws.greengrass.Cli`

Operasi	Deskripsi	Sumber daya
`aws.greengrass#CreateLocalDeployment`	Memungkinkan komponen untuk membuat penyebaran lokal pada perangkat inti.	*
`aws.greengrass#ListLocalDeployments`	Memungkinkan komponen untuk mencantumkan	*

Operasi	Deskripsi	Sumber daya
	penerapan lokal pada perangkat inti.	
`aws.greengrass#GetLocalDeploymentStatus`	Memungkinkan komponen untuk mendapatkan status penyebaran lokal pada perangkat inti.	ID penyebaran lokal, atau * untuk mengizinkan akses ke semua penerapan lokal.
`aws.greengrass#ListComponents`	Memungkinkan komponen untuk daftar komponen pada perangkat inti.	*
`aws.greengrass#GetComponentDetails`	Memungkinkan komponen untuk mendapatkan detail tentang komponen pada perangkat inti.	Nama komponen, seperti`com.example.HelloWorld`, atau * untuk memungkinkan akses ke semua komponen.
`aws.greengrass#RestartComponent`	Memungkinkan komponen untuk me-restart komponen pada perangkat inti.	Nama komponen, seperti`com.example.HelloWorld`, atau * untuk memungkinkan akses ke semua komponen.
`aws.greengrass#StopComponent`	Memungkinkan komponen untuk menghentikan komponen pada perangkat inti.	Nama komponen, seperti`com.example.HelloWorld`, atau * untuk memungkinkan akses ke semua komponen.
`aws.greengrass#CreateDebugPassword`	Mengizinkan komponen menghasilkan kata sandi yang akan digunakan untuk masuk ke komponen konsol debug lokal.	*

Example Contoh kebijakan otorisasi

Contoh kebijakan otorisasi berikut memungkinkan komponen untuk membuat penerapan lokal, melihat semua penerapan dan komponen lokal, dan memulai ulang dan menghentikan komponen bernama. `com.example.HelloWorld`

```
{
  "accessControl": {
    "aws.greengrass.Cli": {
      "com.example.MyLocalManagerComponent:cli:1": {
        "policyDescription": "Allows access to create local deployments and view
 deployments and components.",
        "operations": [
          "aws.greengrass#CreateLocalDeployment",
          "aws.greengrass#ListLocalDeployments",
          "aws.greengrass#GetLocalDeploymentStatus",
          "aws.greengrass#ListComponents",
          "aws.greengrass#GetComponentDetails"
        ],
        "resources": [
          "*"
        ]
      }
    },
    "aws.greengrass.Cli": {
      "com.example.MyLocalManagerComponent:cli:2": {
        "policyDescription": "Allows access to restart and stop the Hello World
 component.",
        "operations": [
          "aws.greengrass#RestartComponent",
          "aws.greengrass#StopComponent"
        ],
        "resources": [
          "com.example.HelloWorld"
        ]
      }
    }
  }
}
```

CreateLocalDeployment

Buat atau perbarui deployment lokal menggunakan resep komponen tertentu, artefak, dan argumen waktu aktif.

Operasi ini menyediakan fungsionalitas yang sama dengan perintah deployment create di Greengrass CLI.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`recipeDirectoryPath`(Python:) `recipe_directory_path`

(Opsional) Jalur absolut ke folder yang berisi file resep komponen.

`artifactDirectoryPath`(Python:) `artifact_directory_path`

(Opsional) Jalur absolut ke folder yang berisi file artefak untuk disertakan dalam penerapan. Folder artefak harus berisi struktur folder berikut:

```
/path/to/artifact/folder/component-name/component-version/artifacts
```

`rootComponentVersionsToAdd`(Python:) `root_component_versions_to_add`

(Opsional) Versi komponen yang akan diinstal pada perangkat inti. Objek ini,`ComponentToVersionMap`, adalah peta yang berisi pasangan kunci-nilai berikut:

`key`

Nama komponen.

`value`

Versi komponen.

`rootComponentsToRemove`(Python:) `root_components_to_remove`

(Opsional) Komponen yang akan dihapus dari perangkat inti. Tentukan daftar di mana setiap entri adalah nama komponen.

`componentToConfiguration`(Python:) `component_to_configuration`

(Opsional) Pembaruan konfigurasi untuk setiap komponen dalam penerapan. Objek ini,`ComponentToConfiguration`, adalah peta yang berisi pasangan kunci-nilai berikut:

`key`

Nama komponen.

`value`

Konfigurasi memperbarui objek JSON untuk komponen. Objek JSON harus memiliki format berikut.

```
{
  "MERGE": {
    "config-key": "config-value"
  },
  "RESET": [
    "path/to/reset/"
  ]
}
```

Untuk informasi selengkapnya tentang pembaruan konfigurasi, lihat Perbarui konfigurasi komponen.

`componentToRunWithInfo`(Python:) `component_to_run_with_info`

(Opsional) Konfigurasi runtime untuk setiap komponen dalam penerapan. Konfigurasi ini mencakup pengguna sistem yang memiliki proses masing-masing komponen dan batas sistem untuk diterapkan pada setiap komponen. Objek ini,`ComponentToRunWithInfo`, adalah peta yang berisi pasangan kunci-nilai berikut:

`key`

Nama komponen.

`value`

Konfigurasi runtime untuk komponen. Jika Anda menghilangkan parameter konfigurasi runtime, perangkat lunak AWS IoT Greengrass Core menggunakan nilai default yang Anda konfigurasikan pada inti Greengrass. Objek ini, `RunWithInfo`, berisi informasi berikut:

`posixUser`(Python:) `posix_user`

(Opsional) Pengguna sistem POSIX dan, secara opsional, grup untuk digunakan untuk menjalankan komponen ini pada perangkat inti Linux. Pengguna, dan grup jika ditentukan, harus ada di setiap perangkat inti Linux. Tentukan pengguna dan grup yang dipisahkan

dengan titik dua (`:`) dalam format berikut: `user:group`. Grup ini opsional. Jika Anda tidak menentukan grup, perangkat lunak AWS IoT Greengrass Core menggunakan grup utama untuk pengguna. Lihat informasi yang lebih lengkap di Konfigurasikan pengguna yang menjalankan komponen.

`windowsUser`(Python:) `windows_user`

(Opsional) Pengguna Windows yang digunakan untuk menjalankan komponen ini pada perangkat inti Windows. Pengguna harus ada di setiap perangkat inti Windows, dan nama serta kata sandinya harus disimpan dalam instance Credentials Manager LocalSystem akun. Lihat informasi yang lebih lengkap di Konfigurasikan pengguna yang menjalankan komponen.

`systemResourceLimits`(Python:) `system_resource_limits`

(Opsional) Batas sumber daya sistem untuk diterapkan pada proses komponen ini. Anda dapat menerapkan batas sumber daya sistem ke komponen Lambda generik dan non-kontainer. Lihat informasi yang lebih lengkap di Konfigurasikan batas sumber daya sistem untuk komponen.

AWS IoT Greengrasssaat ini tidak mendukung fitur ini di perangkat inti Windows.

Objek ini, `SystemResourceLimits`, berisi informasi berikut:

`cpus`

(Opsional) Jumlah maksimum waktu CPU yang dapat digunakan proses komponen ini pada perangkat inti. Total waktu CPU perangkat inti setara dengan jumlah inti CPU perangkat. Misalnya, pada perangkat inti dengan 4 core CPU, Anda dapat mengatur nilai ini 2 untuk membatasi proses komponen ini hingga 50 persen penggunaan setiap inti CPU. Pada perangkat dengan 1 inti CPU, Anda dapat mengatur nilai ini `0.25` untuk membatasi proses komponen ini hingga 25 persen penggunaan CPU. Jika Anda menetapkan nilai ini ke angka yang lebih besar dari jumlah inti CPU, perangkat lunak AWS IoT Greengrass Core tidak membatasi penggunaan CPU komponen.

`memory`

(Opsional) Jumlah maksimum RAM (dalam kilobyte) yang dapat digunakan proses komponen ini pada perangkat inti.

`groupName`(Python:) `group_name`

(Opsional) Nama grup hal yang akan ditargetkan dengan penerapan ini.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`deploymentId`(Python:) `deployment_id`

ID penyebaran lokal yang dibuat permintaan.

ListLocalDeployments

Mendapat status 10 penerapan lokal terakhir.

Operasi ini menyediakan fungsionalitas yang sama dengan perintah daftar penerapan di CLI Greengrass.

Permintaan

Operasi ini tidak memiliki parameter apa pun.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`localDeployments`(Python:) `local_deployments`

Daftar penyebaran lokal. Setiap objek dalam daftar ini adalah `LocalDeployment` objek, yang berisi informasi berikut:

`deploymentId`(Python:) `deployment_id`

ID penyebaran lokal.

`status`

Status penyebaran lokal. Enum ini, `DeploymentStatus`, memiliki nilai-nilai berikut:

- `QUEUED`
- `IN_PROGRESS`
- `SUCCEEDED`
- `FAILED`

GetLocalDeploymentStatus

Mendapat status penyebaran lokal.

Operasi ini menyediakan fungsionalitas yang sama dengan perintah status penerapan di CLI Greengrass.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`deploymentId`(Python:) `deployment_id`

ID penyebaran lokal untuk mendapatkan.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`deployment`

Penyebaran lokal. Objek ini, `LocalDeployment`, berisi informasi berikut:

`deploymentId`(Python:) `deployment_id`

ID penyebaran lokal.

`status`

Status penyebaran lokal. Enum ini, `DeploymentStatus`, memiliki nilai-nilai berikut:

- `QUEUED`
- `IN_PROGRESS`
- `SUCCEEDED`
- `FAILED`

ListComponents

Mendapat nama, versi, status, dan konfigurasi setiap komponen root pada perangkat inti. Komponen root adalah komponen yang Anda tentukan dalam penerapan. Respons ini tidak termasuk komponen yang diinstal sebagai dependensi komponen lain.

Operasi ini menyediakan fungsionalitas yang sama dengan perintah daftar komponen di CLI Greengrass.

Permintaan

Operasi ini tidak memiliki parameter apa pun.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`components`

Daftar komponen root pada perangkat inti. Setiap objek dalam daftar ini adalah `ComponentDetails` objek, yang berisi informasi berikut:

`componentName`(Python:) `component_name`

Nama komponen.

`version`

Versi komponen.

`state`

Keadaan komponen. Keadaan ini dapat menjadi salah satu dari yang berikut:

- `BROKEN`
- `ERRORED`
- `FINISHED`
- `INSTALLED`
- `NEW`
- `RUNNING`
- `STARTING`
- `STOPPING`

`configuration`

Konfigurasi komponen sebagai objek JSON.

GetComponentDetails

Mendapat versi, status, dan konfigurasi komponen pada perangkat inti.

Operasi ini menyediakan fungsionalitas yang sama dengan perintah detail komponen di CLI Greengrass.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`componentName`(Python:) `component_name`

Nama komponen yang akan didapat.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`componentDetails`(Python:) `component_details`

Detail komponen. Objek ini, `ComponentDetails`, berisi informasi berikut:

`componentName`(Python:) `component_name`

Nama komponen.

`version`

Versi komponen.

`state`

Keadaan komponen. Keadaan ini dapat menjadi salah satu dari yang berikut:

- `BROKEN`
- `ERRORED`
- `FINISHED`
- `INSTALLED`
- `NEW`
- `RUNNING`
- `STARTING`

- `STOPPING`

`configuration`

Konfigurasi komponen sebagai objek JSON.

RestartComponent

Memulai ulang komponen pada perangkat inti.

> **Note**
>
> Meskipun Anda dapat me-restart komponen apa pun, kami sarankan Anda hanya me-restart komponen generik.

Operasi ini menyediakan fungsionalitas yang sama dengan perintah restart komponen di CLI Greengrass.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`componentName`(Python:) `component_name`

Nama komponen.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`restartStatus`(Python:) `restart_status`

Status permintaan restart. Status permintaan dapat berupa salah satu dari yang berikut:

- `SUCCEEDED`
- `FAILED`

`message`

Pesan tentang mengapa komponen gagal memulai ulang, jika permintaan gagal.

StopComponent

Menghentikan proses komponen pada perangkat inti.

> **Note**
>
> Meskipun Anda dapat menghentikan komponen apa pun, kami sarankan Anda menghentikan hanya komponen generik.

Operasi ini menyediakan fungsionalitas yang sama dengan perintah stop komponen di CLI Greengrass.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`componentName`(Python:) `component_name`

Nama komponen.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`stopStatus`(Python:) `stop_status`

Status permintaan berhenti. Status permintaan dapat berupa salah satu dari yang berikut:

- `SUCCEEDED`
- `FAILED`

`message`

Pesan tentang mengapa komponen gagal berhenti, jika permintaan gagal.

CreateDebugPassword

Menghasilkan kata sandi acak yang dapat Anda gunakan untuk masuk ke komponen konsol debug lokal. Kata sandi kedaluwarsa 8 jam setelah dibuat.

Operasi ini menyediakan fungsionalitas yang sama dengan get-debug-password perintah di CLI Greengrass.

Permintaan

Operasi ini tidak memiliki parameter apa pun.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`username`

Nama pengguna yang digunakan untuk masuk.

`password`

Kata sandi yang digunakan untuk masuk.

`passwordExpiration`(Python:) `password_expiration`

Waktu ketika kata sandi kedaluwarsa.

`certificateSHA256Hash`(Python:) `certificate_sha256_hash`

Sidik jari SHA-256 untuk sertifikat yang ditandatangani sendiri yang digunakan konsol debug lokal saat HTTPS diaktifkan. Saat Anda membuka konsol debug lokal, gunakan sidik jari ini untuk memverifikasi bahwa sertifikat tersebut sah dan koneksi aman.

`certificateSHA1Hash`(Python:) `certificate_sha1_hash`

Sidik jari SHA-1 untuk sertifikat yang ditandatangani sendiri yang digunakan konsol debug lokal saat HTTPS diaktifkan. Saat Anda membuka konsol debug lokal, gunakan sidik jari ini untuk memverifikasi bahwa sertifikat tersebut sah dan koneksi aman.

Otentikasi dan otorisasi perangkat klien

> **Note**
>
> Fitur ini tersedia untuk v2.6.0 dan yang lebih baru dari komponen inti Greengrass.

Gunakan layanan IPC autentikasi perangkat klien untuk mengembangkan komponen broker lokal khusus di mana perangkat IoT lokal, seperti perangkat klien, dapat terhubung.

Untuk menggunakan operasi IPC ini, sertakan versi 2.2.0 atau yang lebih baru dari komponen autentikasi perangkat klien sebagai dependensi dalam komponen kustom Anda. Anda kemudian dapat menggunakan operasi IPC di komponen kustom Anda untuk melakukan hal berikut:

- Verifikasi identitas perangkat klien yang terhubung ke perangkat inti.
- Buat sesi untuk perangkat klien untuk terhubung ke perangkat inti.
- Verifikasi apakah perangkat klien memiliki izin untuk melakukan tindakan.
- Menerima pemberitahuan saat sertifikat server perangkat inti berputar.

Topik

SDK (Versi Minimum)

Tabel berikut mencantumkan versi minimum AWS IoT Device SDK yang harus Anda gunakan untuk berinteraksi dengan layanan IPC autentikasi perangkat klien.

SDK	Versi minimum	
AWS IoT Device SDKuntuk Java v2	v1.9.3	
AWS IoT Device SDKuntuk Python v2	v1.11.3	
AWS IoT Device SDKuntuk C++ v2	v1.18.3	
AWS IoT Device SDKuntuk JavaScript v2	v1.12.0	

Otorisasi

Untuk menggunakan layanan IPC autentikasi perangkat klien dalam komponen kustom, Anda harus menentukan kebijakan otorisasi yang memungkinkan komponen Anda melakukan operasi ini. Untuk informasi tentang cara menentukan kebijakan otorisasi, lihat Otorisasi komponen untuk melakukan operasi IPC.

Kebijakan otorisasi untuk otentikasi dan otorisasi perangkat klien memiliki properti berikut.

Pengenal layanan IPC: `aws.greengrass.clientdevices.Auth`

Operasi	Deskripsi	Sumber daya
`aws.greengrass#VerifyClientDeviceIdentity`	Memungkinkan komponen untuk memverifikasi identitas perangkat klien.	*
`aws.greengrass#GetClientDeviceAuthToken`	Memungkinkan komponen untuk memvalidasi kredensi perangkat klien dan membuat sesi untuk perangkat klien tersebut.	*
`aws.greengrass#AuthorizeClientDeviceAction`	Memungkinkan komponen untuk memverifikasi apakah perangkat klien memiliki izin untuk melakukan tindakan.	*
`aws.greengrass#SubscribeToCertificateUpdates`	Memungkinkan komponen untuk menerima pemberitahuan ketika sertifikat server perangkat inti berputar.	*
*	Memungkinkan komponen untuk melakukan semua operasi layanan IPC autentikasi perangkat klien.	*

Contoh kebijakan otorisasi

Anda dapat mereferensikan contoh kebijakan otorisasi berikut untuk membantu Anda mengonfigurasi kebijakan otorisasi untuk komponen Anda.

Example Contoh kebijakan otorisasi

Contoh kebijakan otorisasi berikut memungkinkan komponen untuk melakukan semua operasi IPC autentikasi perangkat klien.

```
{
  "accessControl": {
    "aws.greengrass.clientdevices.Auth": {
      "com.example.MyLocalBrokerComponent:clientdevices:1": {
        "policyDescription": "Allows access to authenticate and authorize client
 devices.",
        "operations": [
          "aws.greengrass#VerifyClientDeviceIdentity",
          "aws.greengrass#GetClientDeviceAuthToken",
          "aws.greengrass#AuthorizeClientDeviceAction",
          "aws.greengrass#SubscribeToCertificateUpdates"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
}
```

VerifyClientDeviceIdentity

Verifikasi identitas perangkat klien. Operasi ini memverifikasi apakah perangkat klien adalah AWS IoT hal yang valid.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`credential`

Kredensi perangkat klien. Objek ini, `ClientDeviceCredential`, berisi informasi berikut:

`clientDeviceCertificate`(Python:) `client_device_certificate`

Sertifikat perangkat X.509 perangkat klien.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`isValidClientDevice`(Python:) `is_valid_client_device`

Apakah identitas perangkat klien valid.

GetClientDeviceAuthToken

Memvalidasi kredensi perangkat klien dan membuat sesi untuk perangkat klien. Operasi ini mengembalikan token sesi yang dapat Anda gunakan dalam permintaan berikutnya untuk mengotorisasi tindakan perangkat klien.

Agar berhasil menghubungkan perangkat klien, komponen autentikasi perangkat klien harus memberikan `mqtt:connect` izin untuk ID klien yang digunakan perangkat klien.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`credential`

Kredensi perangkat klien. Objek ini, `CredentialDocument`, berisi informasi berikut:

`mqttCredential`(Python:) `mqtt_credential`

Kredensi MQTT perangkat klien. Tentukan ID klien dan sertifikat yang digunakan perangkat klien untuk terhubung. Objek ini, `MQTTCredential`, berisi informasi berikut:

`clientId`(Python:) `client_id`

ID klien yang digunakan untuk terhubung.

`certificatePem`(Python:) `certificate_pem`

Sertifikat perangkat X.509 yang digunakan untuk menghubungkan.

`username`

> **Note**
>
> Properti ini saat ini tidak digunakan.

`password`

> **Note**
>
> Properti ini saat ini tidak digunakan.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`clientDeviceAuthToken`(Python:) `client_device_auth_token`

Token sesi untuk perangkat klien. Anda dapat menggunakan token sesi ini dalam permintaan berikutnya untuk mengotorisasi tindakan perangkat klien ini.

AuthorizeClientDeviceAction

Verifikasi apakah perangkat klien memiliki izin untuk melakukan tindakan pada sumber daya. Kebijakan otorisasi perangkat klien menentukan izin yang dapat dilakukan perangkat klien saat terhubung ke perangkat inti. Anda menentukan kebijakan otorisasi perangkat klien saat mengonfigurasi komponen autentikasi perangkat klien.

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`clientDeviceAuthToken`(Python:) `client_device_auth_token`

Token sesi untuk perangkat klien.

`operation`

Operasi untuk mengotorisasi.

`resource`

Sumber daya tempat perangkat klien melakukan operasi.

Respons

Tanggapan operasi ini memiliki informasi berikut:

`isAuthorized`(Python:) `is_authorized`

Apakah perangkat klien berwenang untuk melakukan operasi pada sumber daya.

SubscribeToCertificateUpdates

Berlangganan untuk menerima sertifikat server baru perangkat inti setiap kali berputar. Ketika sertifikat server berputar, broker harus memuat ulang menggunakan sertifikat server baru.

Komponen autentikasi perangkat klien memutar sertifikat server setiap 7 hari secara default. Anda dapat mengkonfigurasi interval rotasi antara 2 dan 10 hari.

Operasi ini adalah operasi berlangganan di mana Anda berlangganan aliran pesan peristiwa. Untuk menggunakan operasi ini, tentukan bagian yang menangani respons aliran dengan fungsi yang menangani pesan peristiwa, kesalahan, dan penutupan aliran. Untuk informasi selengkapnya, lihat Berlangganan pengaliran peristiwa IPC.

Jenis pesan peristiwa: `CertificateUpdateEvent`

Permintaan

Permintaan operasi ini memiliki parameter berikut:

`certificateOptions`(Python:) `certificate_options`

Jenis pembaruan sertifikat untuk berlangganan. Objek ini, `CertificateOptions`, berisi informasi berikut:

`certificateType`(Python:) `certificate_type`

Jenis pembaruan sertifikat untuk berlangganan. Pilih opsi berikut:

- `SERVER`

Respons

Tanggapan operasi ini memiliki informasi berikut:

`messages`

Aliran pesan. Objek ini, `CertificateUpdateEvent`, berisi informasi berikut:

`certificateUpdate`(Python:) `certificate_update`

Informasi tentang sertifikat baru. Objek ini, `CertificateUpdate`, berisi informasi berikut:

`certificate`

Sertifikat.

`privateKey`(Python:) `private_key`

Kunci pribadi sertifikat.

`publicKey`(Python:) `public_key`

Kunci publik sertifikat.

`caCertificates`(Python:) `ca_certificates`

Daftar sertifikat otoritas sertifikat (CA) dalam rantai sertifikat CA sertifikat.

Berinteraksilah dengan perangkat IoT lokal

Perangkat klien adalah perangkat IoT lokal yang terhubung ke dan berkomunikasi dengan perangkat inti Greengrass melalui MQTT. Anda dapat menghubungkan perangkat klien ke perangkat inti untuk melakukan hal berikut:

- Berinteraksi dengan pesan MQTT dalam komponen Greengrass.
- Merelai pesan dan data antara perangkat klien dan AWS IoT Core.
- Berinteraksi dengan bayangan perangkat klien dalam komponen Greengrass.
- Menyinkronkan bayangan perangkat klien dengan AWS IoT Core.

Untuk terhubung ke perangkat inti, perangkat klien dapat menggunakan penemuan cloud Perangkat klien dapat tersambung ke layanan cloud AWS IoT Greengrass untuk mengambil informasi tentang perangkat inti yang dapat ia hubungkan. Kemudian, perangkat klien tersebut dapat terhubung ke perangkat inti untuk memproses pesannya dan menyinkronkan datanya dengan layanan cloud AWS IoT Core.

Anda dapat mengikuti tutorial yang membahas cara mengonfigurasi perangkat inti untuk terhubung dan berkomunikasi dengan objek AWS IoT. Tutorial ini juga mengeksplorasi bagaimana mengembangkan komponen Greengrass kustom yang berinteraksi dengan perangkat klien. Lihat informasi yang lebih lengkap di Tutorial: Berinteraksi dengan perangkat IoT lokal melalui MQTT.

Topik

Komponen perangkat klien yang disediakan oleh AWS

AWS IoT Greengrass menyediakan komponen publik berikut yang dapat Anda deploy ke perangkat inti. Komponen ini memungkinkan perangkat klien untuk terhubung dan berkomunikasi dengan perangkat inti.

> **Note**
>
> Beberapa komponen AWS yang disediakan bergantung pada versi minor spesifik dari inti Greengrass. Karena ketergantungan ini, Anda perlu memperbarui komponen ini saat memperbarui inti Greengrass ke versi minor yang baru. Untuk informasi tentang versi spesifik dari inti yang masing-masing komponen bergantung padanya, lihat topik komponen yang sesuai. Untuk informasi selengkapnya terkait cara memperbarui inti, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Ketika komponen memiliki tipe komponen generik dan Lambda, versi komponen saat ini adalah tipe generik dan versi komponen sebelumnya adalah tipe Lambda.

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Auth perangkat klien	Memungkinkan perangkat IoT lokal, yang disebut perangkat klien, untuk terhubung ke perangkat inti.	Plugin	Linux, Windows	Ya
Detektor IP	Melaporkan informasi konektivitas broker MQTT	Plugin	Linux, Windows	Ya

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
	ke AWS IoT Greengrass, sehingga perangkat klien dapat menemukan cara untuk terhubung.			
Jembatan MQTT	Relai pesan MQTT antara perangkat klien, publikasi/berlangganan AWS IoT Greengrass lokal, dan AWS IoT Core.	Plugin	Linux, Windows	Ya
MQTT 3.1.1 broker (Moquette)	Menjalankan broker MQTT 3.1.1 yang menangani pesan antara perangkat klien dan perangkat inti.	Plugin	Linux, Windows	Ya

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Pialang MQTT 5 (EMQX)	Menjalankan broker MQTT 5 yang menangani pesan antara perangkat klien dan perangkat inti.	Generik	Linux, Windows	Tidak
Pengelola bayangan	Memungkinkan interaksi dengan bayangan pada perangkat inti. Ia mengelola penyimpanan dokumen bayangan dan juga sinkronisasi keadaan bayangan lokal dengan layanan Bayangan Perangkat AWS IoT.	Plugin	Linux, Windows	Ya

Hubungkan perangkat klien ke perangkat inti

Anda dapat mengonfigurasi Penemuan cloud untuk menghubungkan perangkat klien ke perangkat inti. Bila Anda mengonfigurasi penemuan cloud, perangkat klien dapat tersambung ke layanan cloud AWS IoT Greengrass untuk mengambil informasi tentang perangkat inti yang dapat ia hubungkan. Kemudian, perangkat klien dapat mencoba untuk menyambung ke setiap perangkat inti sampai berhasil terhubung.

Untuk menggunakan penemuan cloud, Anda harus melakukan hal berikut:

- Kaitkan perangkat klien ke perangkat inti tempat mereka dapat terhubung.
- Tentukan titik akhir broker MQTT di mana perangkat klien dapat terhubung ke setiap perangkat inti.
- Deploy komponen ke perangkat inti yang memungkinkan dukungan untuk perangkat klien.

  Anda juga dapat menggunakan komponen opsional untuk melakukan hal berikut:

  - Relai pesan antara perangkat klien, komponen Greengrass, dan layanan cloud AWS IoT Core.
  - Secara otomatis kelola titik akhir broker MQTT perangkat inti untuk Anda.
  - Kelola bayangan perangkat klien lokal dan sinkronkan bayangan dengan layanan AWS IoT Core cloud.

Anda juga harus meninjau dan memperbarui AWS IoT kebijakan perangkat inti untuk memverifikasi bahwa ia memiliki izin yang diperlukan untuk menghubungkan perangkat klien. Untuk informasi selengkapnya, lihat Persyaratan.

Setelah Anda mengonfigurasi penemuan cloud, Anda dapat menguji komunikasi antara perangkat klien dan perangkat inti. Untuk informasi selengkapnya, lihat Uji komunikasi perangkat klien.

Topik

- Pilih broker MQTT
- Menghubungkan perangkat klien ke perangkatAWS IoT Greengrass Core dengan broker MQTT
- Uji komunikasi perangkat klien
- Greengrass discovery RESTful API

Persyaratan

Untuk menghubungkan perangkat klien ke perangkat inti, Anda harus memiliki yang berikut:

- Perangkat inti harus menjalankan Greengrass nucleus v2.2.0 atau yang lebih baru.
- Peran layanan Greengrass yang terkait AWS IoT Greengrass dengan untuk Akun AWS Anda di AWS Wilayah tempat perangkat inti beroperasi. Untuk informasi selengkapnya, lihat Konfigurasikan peran layanan Greengrass.
- Kebijakan perangkat inti AWS IoT ini harus memungkinkan izin berikut:
  - `greengrass:PutCertificateAuthorities`
  - `greengrass:VerifyClientDeviceIdentity`
  - `greengrass:VerifyClientDeviceIoTCertificateAssociation`
  - `greengrass:GetConnectivityInfo`
  - `greengrass:UpdateConnectivityInfo` – (Opsional) Izin ini diperlukan untuk menggunakan komponen detektor IP, yang melaporkan informasi konektivitas jaringan perangkat inti ke layanan cloud AWS IoT Greengrass.
  - `iot:GetThingShadow`,`iot:UpdateThingShadow`, dan `iot:DeleteThingShadow` — (Opsional) Izin ini diperlukan untuk menggunakan komponen shadow manager untuk menyinkronkan bayangan perangkat klien denganAWS IoT Core. Fitur ini membutuhkan Greengrass nucleus v2.6.0 atau yang lebih baru, shadow manager v2.2.0 atau yang lebih baru, dan MQTT bridge v2.2.0 atau yang lebih baru.

  Untuk informasi selengkapnya, lihat Konfigurasikan kebijakan AWS IoT hal.

> **Note**
>
> Jika Anda menggunakan AWS IoT kebijakan default saat menginstal perangkat lunak AWS IoT Greengrass Core, perangkat inti memiliki AWS IoT kebijakan yang memungkinkan akses ke semua AWS IoT Greengrass tindakan (`greengrass:*`).

- AWS IoThal-hal yang dapat Anda hubungkan sebagai perangkat klien. Untuk informasi lebih lanjut, lihat Buat AWS IoT sumber daya di AWS IoT Core Panduan Developer.
- Perangkat klien harus terhubung menggunakan ID klien. ID klien adalah nama benda. Tidak ada ID klien lain yang akan diterima.
- AWS IoTKebijakan setiap perangkat klien harus mengizinkan `greengrass:Discover` izin tersebut. Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat klien.

Topik

- Konfigurasikan peran layanan Greengrass
- Konfigurasikan kebijakan AWS IoT hal

Konfigurasikan peran layanan Greengrass

Peran layanan Greengrass adalah AWS Identity and Access Management peran layanan (IAM) yang mengotorisasi AWS IoT Greengrass untuk mengakses sumber daya dari AWS layanan atas nama Anda. Peran ini memungkinkan AWS IoT Greengrass untuk memverifikasi identitas perangkat klien dan mengelola informasi konektivitas perangkat inti.

Jika sebelumnya Anda belum menyiapkan peran layanan Greengrass di Wilayah ini, Anda harus mengaitkan peran layanan Greengrass untuk Anda di Wilayah ini. AWS IoT Greengrass Akun AWS

Saat Anda menggunakan halaman Konfigurasi penemuan perangkat inti di AWS IoT Greengrasskonsol, siapkan AWS IoT Greengrass peran layanan Greengrass untuk Anda. Jika tidak, Anda dapat mengaturnya secara manual menggunakan AWS IoTkonsol atau AWS IoT Greengrass API.

Di bagian ini, Anda memeriksa apakah peran layanan Greengrass sudah diatur. Jika tidak disiapkan, Anda membuat peran layanan Greengrass baru untuk dikaitkan AWS IoT Greengrass dengan Anda di Wilayah ini. Akun AWS

Konfigurasikan peran layanan Greengrass (konsol)

1. Periksa apakah peran layanan Greengrass dikaitkan AWS IoT Greengrass dengan untuk Anda di Wilayah ini. Akun AWS Lakukan hal-hal berikut:

   a. Navigasikan ke konsol AWS IoT tersebut.

   b. Di panel navigasi, pilih Pengaturan.

c. Di bagian peran layanan Greengrass, temukan Peran layanan saat ini untuk melihat apakah peran layanan Greengrass terkait.

   Jika Anda memiliki peran layanan Greengrass yang terkait, Anda memenuhi persyaratan ini untuk menggunakan komponen detektor IP. Loncat ke Konfigurasikan kebijakan AWS IoT hal.

2. Jika peran layanan Greengrass tidak terkait AWS IoT Greengrass dengan peran Akun AWS Anda di Wilayah ini, buat peran layanan Greengrass dan kaitkan. Lakukan hal-hal berikut:

   a. Arahkan ke konsol IAM.

   b. Pilih Peran.

   c. Pilih Buat peran.

   d. Pada halaman Buat peran, lakukan hal berikut:

      i. Di bawah Jenis entitas tepercaya, pilih Layanan AWS.

      ii. Di bawah Kasus penggunaan, Gunakan kasus untuk lainnya Layanan AWS, pilih Greengrass, pilih Greengrass. Opsi ini menentukan untuk menambahkan AWS IoT Greengrass sebagai entitas tepercaya yang dapat mengambil peran ini.

      iii. Pilih Berikutnya.

      iv. Di bawah Kebijakan izin, pilih yang akan dilampirkan AWSGreengrassResourceAccessRolePolicyke peran.

      v. Pilih Berikutnya.

      vi. Di Nama peran, masukkan nama untuk peran tersebut, seperti**`Greengrass_ServiceRole`**.

      vii. Pilih Buat peran.

   e. Navigasikan ke konsol AWS IoT tersebut.

   f. Di panel navigasi, pilih Pengaturan.

   g. Di bagian peran layanan Greengrass, pilih Lampirkan peran.

   h. Dalam Perbarui modal peran layanan Greengrass, pilih peran IAM yang Anda buat, lalu pilih Lampirkan peran.

Konfigurasikan peran layanan Greengrass () AWS CLI

1. Periksa apakah peran layanan Greengrass dikaitkan AWS IoT Greengrass dengan untuk Anda di Wilayah ini. Akun AWS

   ```
   aws greengrassv2 get-service-role-for-account
   ```

   Jika peran layanan Greengrass dikaitkan, operasi mengembalikan respons yang berisi informasi tentang peran tersebut.

   Jika Anda memiliki peran layanan Greengrass yang terkait, Anda memenuhi persyaratan ini untuk menggunakan komponen detektor IP. Loncat ke Konfigurasikan kebijakan AWS IoT hal.

2. Jika peran layanan Greengrass tidak terkait AWS IoT Greengrass dengan peran Akun AWS Anda di Wilayah ini, buat peran layanan Greengrass dan kaitkan. Lakukan hal-hal berikut:

   a. Buat peran dengan kebijakan kepercayaan yang memungkinkan AWS IoT Greengrass untuk mengambil peran. Contoh ini menciptakan peran bernama `Greengrass_ServiceRole`, tetapi Anda dapat menggunakan nama yang berbeda. Kami menyarankan Anda juga menyertakan `aws:SourceArn` dan kunci konteks kondisi `aws:SourceAccount` global dalam kebijakan kepercayaan Anda untuk membantu mencegah masalah keamanan wakil yang membingungkan. Kunci konteks kondisi membatasi akses untuk mengizinkan hanya permintaan yang berasal dari akun tertentu dan ruang kerja Greengrass. Untuk informasi lebih lanjut tentang masalah wakil yang membingungkan, lihatCross-service bingung wakil pencegahan.

      Linux or Unix

      ```
      aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
      document '{
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "greengrass.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
              "ArnLike": {
                "aws:SourceArn": "arn:aws:greengrass:region:account-id:*"
      ```

```
        },
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        }
      }
    }
  ]
}'
```

Windows Command Prompt (CMD)

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
document "{\\"Version\\":\\"2012-10-17\\",\\"Statement\\":[{\\"Effect\
\":\\"Allow\\",\\"Principal\\":{\\"Service\\":\\"greengrass.amazonaws.com
\\"},\\"Action\\":\\"sts:AssumeRole\\",\\"Condition\\":{\\"ArnLike\\":
{\\"aws:SourceArn\\":\\"arn:aws:greengrass:region:account-id:*\\"},\
\"StringEquals\\":{\\"aws:SourceAccount\\":\\"account-id\\"}}}]}"
```

PowerShell

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "greengrass.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:greengrass:region:account-id:*"
        },
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        }
      }
    }
  ]
}'
```

b. Salin peran ARN dari metadata peran dalam output. Anda menggunakan ARN untuk mengasosiasikan peran dengan akun Anda.

c. 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
```

d. Kaitkan peran layanan Greengrass dengan untuk Anda. AWS IoT Greengrass Akun AWS Ganti *`role-arn`* dengan ARN dari peran layanan.

```
aws greengrassv2 associate-service-role-to-account --role-arn role-arn
```

Operasi mengembalikan respon berikut jika berhasil.

```
{
  "associatedAt": "timestamp"
}
```

Konfigurasikan kebijakan AWS IoT hal

Perangkat inti menggunakan sertifikat perangkat X.509 untuk mengotorisasi koneksi ke. AWS Anda melampirkan AWS IoT kebijakan ke sertifikat perangkat untuk menentukan izin untuk perangkat inti. Lihat informasi yang lebih lengkap di Kebijakan AWS IoT untuk operasi bidang data dan Kebijakan AWS IoT minimal untuk mendukung perangkat klien.

Untuk menghubungkan perangkat klien ke perangkat inti, AWS IoT kebijakan perangkat inti harus mengizinkan izin berikut:

- `greengrass:PutCertificateAuthorities`
- `greengrass:VerifyClientDeviceIdentity`
- `greengrass:VerifyClientDeviceIoTCertificateAssociation`
- `greengrass:GetConnectivityInfo`
- `greengrass:UpdateConnectivityInfo` – (Opsional) Izin ini diperlukan untuk menggunakan komponen detektor IP, yang melaporkan informasi konektivitas jaringan perangkat inti ke layanan cloud AWS IoT Greengrass.

- `iot:GetThingShadow`,`iot:UpdateThingShadow`, dan `iot:DeleteThingShadow` — (Opsional) Izin ini diperlukan untuk menggunakan komponen shadow manager untuk menyinkronkan bayangan perangkat klien denganAWS IoT Core. Fitur ini membutuhkan Greengrass nucleus v2.6.0 atau yang lebih baru, shadow manager v2.2.0 atau yang lebih baru, dan MQTT bridge v2.2.0 atau yang lebih baru.

Di bagian ini, Anda meninjau AWS IoT untuk perangkat inti Anda dan menambahkan izin yang diperlukan yang hilang. Jika Anda menggunakan Penginstal perangkat lunak inti AWS IoT Greengrass untuk menyediakan sumber daya, perangkat inti Anda memiliki kebijakan AWS IoT yang mengizinkan akses ke semua tindakan AWS IoT Greengrass (`greengrass:*`). Dalam hal ini, Anda harus memperbarui kebijakan AWS IoT hanya jika Anda berencana untuk men-deploy komponen manajer bayangan untuk menyinkronkan bayangan perangkat dengan AWS IoT Core. Jika tidak, Anda dapat melewati bagian ini.

Konfigurasikan kebijakan AWS IoT hal (konsol)

1. Di menu navigasi konsol AWS IoT Greengrass tersebut, pilih Perangkat inti.
2. Pada halaman Perangkat inti, pilih perangkat inti yang akan diperbarui.
3. Pada halaman detail perangkat inti, pilih tautan ke Objek perangkat inti. Tautan ini membuka halaman rincian hal di AWS IoT konsol.
4. Pada halaman detail objek, pilih Sertifikat.
5. Di tab Sertifikat, pilih sertifikat aktif objek.
6. Pada halaman detail sertifikat, pilih Kebijakan.
7. Di tab Kebijakan, pilih kebijakan AWS IoT yang akan ditinjau dan diperbarui. Anda dapat menambahkan izin yang diperlukan untuk kebijakan yang dilampirkan ke sertifikat aktif perangkat inti.

> **Note**
>
> Jika Anda menggunakan Penginstal perangkat lunak inti AWS IoT Greengrass untuk menyediakan sumber daya, Anda memiliki dua kebijakan AWS IoT. Kami menyarankan Anda memilih kebijakan yang diberi nama GreengrassV2IoTThingPolicy, jika ada. Perangkat inti yang Anda buat dengan penginstal cepat menggunakan nama kebijakan ini secara default. Jika Anda menambahkan izin untuk kebijakan ini, Anda juga memberikan izin ini ke perangkat inti lain yang menggunakan kebijakan ini.

8. Dalam ikhtisar kebijakan, pilih Edit versi aktif.
9. Tinjau kebijakan untuk izin yang diperlukan, dan tambahkan izin yang diperlukan yang hilang.

   - `greengrass:PutCertificateAuthorities`
   - `greengrass:VerifyClientDeviceIdentity`
   - `greengrass:VerifyClientDeviceIoTCertificateAssociation`
   - `greengrass:GetConnectivityInfo`
   - `greengrass:UpdateConnectivityInfo` – (Opsional) Izin ini diperlukan untuk menggunakan komponen detektor IP, yang melaporkan informasi konektivitas jaringan perangkat inti ke layanan cloud AWS IoT Greengrass.
   - `iot:GetThingShadow`,`iot:UpdateThingShadow`, dan `iot:DeleteThingShadow` — (Opsional) Izin ini diperlukan untuk menggunakan komponen shadow manager untuk menyinkronkan bayangan perangkat klien denganAWS IoT Core. Fitur ini membutuhkan Greengrass nucleus v2.6.0 atau yang lebih baru, shadow manager v2.2.0 atau yang lebih baru, dan MQTT bridge v2.2.0 atau yang lebih baru.
10. (Opsional) Untuk mengizinkan perangkat inti menyinkronkan bayanganAWS IoT Core, tambahkan pernyataan berikut ke kebijakan. Jika Anda berencana untuk berinteraksi dengan bayangan perangkat klien, tetapi tidak menyinkronkannyaAWS IoT Core, lewati langkah ini. Ganti *`wilayah`* dan *`account-id`* dengan Wilayah yang Anda gunakan dan angka Akun AWS Anda.

    - Contoh pernyataan ini memungkinkan akses ke bayangan perangkat semua objek. Untuk mengikuti praktik keamanan terbaik, Anda dapat membatasi akses ke hanya perangkat inti dan perangkat klien yang tersambung ke perangkat inti. Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk mendukung perangkat klien.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:GetThingShadow",
    "iot:UpdateThingShadow",
    "iot:DeleteThingShadow"
  ],
  "Resource": [
    "arn:aws:iot:region:account-id:thing/*"
  ]
```

```
}
```

Setelah Anda menambahkan pernyataan ini, kumpulan dokumen kebijakan mungkin terlihat serupa dengan contoh berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "greengrass:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:DeleteThingShadow"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:thing/*"
      ]
    }
  ]
}
```

11. Untuk menetapkan versi kebijakan baru sebagai versi aktif, di bawah Status versi Kebijakan, pilih Setel versi yang diedit sebagai versi aktif untuk kebijakan ini.
12. Pilih Simpan sebagai versi baru.

Konfigurasikan kebijakan AWS IoT hal (AWS CLI)

1. Buat daftar prinsipal untuk hal perangkat inti. AWS IoT Prinsipal hal dapat berupa sertifikat perangkat X.509 atau identifikasi lainnya. Jalankan perintah berikut, dan ganti *MyGreengrassCore*dengan nama perangkat inti.

   ```
   aws iot list-thing-principals --thing-name MyGreengrassCore
   ```

   Operasi mengembalikan respons yang mencantumkan prinsipal hal perangkat inti.

   ```
   {
       "principals": [
           "arn:aws:iot:us-west-2:123456789012:cert/certificateId"
       ]
   }
   ```

2. Identifikasi sertifikat aktif perangkat inti. Jalankan perintah berikut, dan ganti *CertificateId* dengan ID setiap sertifikat dari langkah sebelumnya hingga Anda menemukan sertifikat aktif. ID sertifikat adalah string heksadesimal di akhir sertifikat ARN. `--query`Argumen menentukan untuk output hanya status sertifikat.

   ```
   aws iot describe-certificate --certificate-id certificateId --query
    'certificateDescription.status'
   ```

   Operasi mengembalikan status sertifikat sebagai string. Misalnya, jika sertifikat aktif, output `"ACTIVE"` operasi ini.

3. Buat daftar AWS IoT kebijakan yang dilampirkan pada sertifikat. Jalankan perintah berikut, dan ganti sertifikat ARN dengan ARN sertifikat.

   ```
   aws iot list-principal-policies --principal arn:aws:iot:us-
   west-2:123456789012:cert/certificateId
   ```

   Operasi mengembalikan respons yang mencantumkan AWS IoT kebijakan yang dilampirkan pada sertifikat.

   ```
   {
       "policies": [
           {
   ```

```
            "policyName":
 "GreengrassTESCertificatePolicyMyGreengrassCoreTokenExchangeRoleAlias",
            "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassTESCertificatePolicyMyGreengrassCoreTokenExchangeRoleAlias"
        },
        {
            "policyName": "GreengrassV2IoTThingPolicy",
            "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy"
        }
    ]
}
```

4. Pilih kebijakan yang akan dilihat dan diperbarui.

> **Note**
>
> Jika Anda menggunakan Penginstal perangkat lunak inti AWS IoT Greengrass untuk menyediakan sumber daya, Anda memiliki dua kebijakan AWS IoT. Kami menyarankan Anda memilih kebijakan yang diberi nama GreengrassV2IoTThingPolicy, jika ada. Perangkat inti yang Anda buat dengan penginstal cepat menggunakan nama kebijakan ini secara default. Jika Anda menambahkan izin untuk kebijakan ini, Anda juga memberikan izin ini ke perangkat inti lain yang menggunakan kebijakan ini.

5. Dapatkan dokumen kebijakan. Jalankan perintah berikut, dan ganti *GreenGrassV2IoT ThingPolicy* dengan nama kebijakan.

```
aws iot get-policy --policy-name GreengrassV2IoTThingPolicy
```

Operasi mengembalikan respons yang berisi dokumen kebijakan dan informasi lain tentang kebijakan tersebut. Dokumen kebijakan adalah objek JSON yang diserialisasikan sebagai string.

```
{
    "policyName": "GreengrassV2IoTThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
    "policyDocument": "{\
  \\"Version\\": \\"2012-10-17\\",\
  \\"Statement\\": [\
    {\
      \\"Effect\\": \\"Allow\\",\
```

```
      \\"Action\\": [\
                \\"iot:Connect\\",\
                \\"iot:Publish\\",\
                \\"iot:Subscribe\\",\
                \\"iot:Receive\\",\
                \\"greengrass:*\\"\
],\
      \\"Resource\\": \\"*\\"\
    }\
  ]\
}",
    "defaultVersionId": "1",
    "creationDate": "2021-02-05T16:03:14.098000-08:00",
    "lastModifiedDate": "2021-02-05T16:03:14.098000-08:00",
    "generationId":
 "f19144b798534f52c619d44f771a354f1b957dfa2b850625d9f1d0fde530e75f"
}
```

6. Gunakan konverter online atau alat lain untuk mengonversi string dokumen kebijakan menjadi objek JSON, lalu simpan ke file bernama`iot-policy.json`.

   Misalnya, jika Anda menginstal alat jq, Anda dapat menjalankan perintah berikut untuk mendapatkan dokumen kebijakan, mengubahnya menjadi objek JSON, dan menyimpan dokumen kebijakan sebagai objek JSON.

   ```
   aws iot get-policy --policy-name GreengrassV2IoTThingPolicy --query
    'policyDocument' | jq fromjson >> iot-policy.json
   ```

7. Tinjau kebijakan untuk izin yang diperlukan, dan tambahkan izin yang diperlukan yang hilang.

   Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuka file.

   ```
   nano iot-policy.json
   ```

   - `greengrass:PutCertificateAuthorities`
   - `greengrass:VerifyClientDeviceIdentity`
   - `greengrass:VerifyClientDeviceIoTCertificateAssociation`
   - `greengrass:GetConnectivityInfo`

- `greengrass:UpdateConnectivityInfo` – (Opsional) Izin ini diperlukan untuk menggunakan komponen detektor IP, yang melaporkan informasi konektivitas jaringan perangkat inti ke layanan cloud AWS IoT Greengrass.
- `iot:GetThingShadow`,`iot:UpdateThingShadow`, dan `iot:DeleteThingShadow` — (Opsional) Izin ini diperlukan untuk menggunakan komponen shadow manager untuk menyinkronkan bayangan perangkat klien denganAWS IoT Core. Fitur ini membutuhkan Greengrass nucleus v2.6.0 atau yang lebih baru, shadow manager v2.2.0 atau yang lebih baru, dan MQTT bridge v2.2.0 atau yang lebih baru.

8. Simpan perubahan sebagai versi baru kebijakan. Jalankan perintah berikut, dan ganti *`GreenGrassV2IoT ThingPolicy`* dengan nama kebijakan.

```
aws iot create-policy-version --policy-name GreengrassV2IoTThingPolicy --policy-
document file://iot-policy.json --set-as-default
```

Operasi mengembalikan respon mirip dengan contoh berikut jika berhasil.

```
{
    "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
    "policyDocument": "{\
  \\"Version\\": \\"2012-10-17\\",\
  \\"Statement\\": [\
    {\
      \\"Effect\\": \\"Allow\\",\
      \\"Action\\": [\
\\t\\t\\"iot:Connect\\",\
\\t\\t\\"iot:Publish\\",\
\\t\\t\\"iot:Subscribe\\",\
\\t\\t\\"iot:Receive\\",\
\\t\\t\\"greengrass:*\\"\
      ],\
      \\"Resource\\": \\"*\\"\
    }\
  ]\
}",
    "policyVersionId": "2",
    "isDefaultVersion": true
}
```

Komponen Greengrass untuk dukungan perangkat klien

> **Important**
>
> Perangkat inti harus menjalankan inti Greengrass v2.2.0 atau yang lebih baru untuk mendukung perangkat klien.

Untuk memungkinkan perangkat klien untuk terhubung dan berkomunikasi dengan perangkat inti, Anda men-deploy komponen Greengrass berikut ke perangkat inti:

- Auth perangkat klien (`aws.greengrass.clientdevices.Auth`)

  Deploy komponen auth perangkat klien untuk mengautentikasi perangkat klien dan mengotorisasi tindakan perangkat klien. Komponen ini memungkinkan objek AWS IoT Anda untuk terhubung ke perangkat inti.

  Komponen ini memerlukan beberapa konfigurasi untuk menggunakannya. Anda harus menentukan grup perangkat klien dan operasi yang setiap grup diotorisasi untuk melakukan, seperti untuk menghubungkan dan berkomunikasi melalui MQTT. Untuk informasi lebih lanjut, lihat konfigurasi komponen auth perangkat klien.
- MQTT 3.1.1 broker (Moquette) (`aws.greengrass.clientdevices.mqtt.Moquette`)

  Menyebarkan komponen broker Moquette MQTT untuk menjalankan broker MQTT ringan. Broker Moquette MQTT patuh pada MQTT 3.1.1 dan mencakup dukungan lokal untuk QoS 0, QoS 1, QoS 2, pesan yang dipertahankan, pesan kehendak terakhir, dan langganan terus-menerus.

  Anda tidak diharuskan mengonfigurasi komponen ini untuk menggunakannya. Namun, Anda dapat mengonfigurasi port di mana komponen ini mengoperasikan broker MQTT. Secara default, ia menggunakan port 8883.
- Pialang MQTT 5 (EMQX) (`aws.greengrass.clientdevices.mqtt.EMQX`)

  > **Note**
  >
  > Untuk menggunakan broker EMQX MQTT 5, Anda harus menggunakan Greengrass nucleus v2.6.0 atau yang lebih baru dan auth perangkat klien v2.2.0 atau yang lebih baru.

Menyebarkan komponen broker EMQX MQTT untuk menggunakan fitur MQTT 5.0 dalam komunikasi antara perangkat klien dan perangkat inti. Broker EMQX MQTT sesuai dengan MQTT 5.0 dan mencakup dukungan untuk interval kedaluwarsa sesi dan pesan, properti pengguna, langganan bersama, alias topik, dan banyak lagi.

Anda tidak diharuskan mengonfigurasi komponen ini untuk menggunakannya. Namun, Anda dapat mengonfigurasi port di mana komponen ini mengoperasikan broker MQTT. Secara default, ia menggunakan port 8883.

- Jembatan MQTT (`aws.greengrass.clientdevices.mqtt.Bridge`)

  (Opsional) Deploy komponen jembatan MQTT untuk merelai pesan antar perangkat klien (MQTT lokal), penerbitan/berlangganan lokal, dan MQTT AWS IoT Core. Konfigurasikan komponen ini untuk menyinkronkan perangkat klien dengan AWS IoT Core dan berinteraksi dengan perangkat klien dari komponen Greengrass.

  Komponen ini memerlukan konfigurasi untuk digunakan. Anda harus menentukan pemetaan topik di mana komponen ini merelai pesan. Untuk informasi lebih lanjut, lihat konfigurasi komponen jembatan MQTT.

- Detektor IP (`aws.greengrass.clientdevices.IPDetector`)

  (Opsional) Deploy komponen detektor IP untuk secara otomatis melaporkan titik akhir broker MQTT perangkat inti ke layanan cloud AWS IoT Greengrass. Anda tidak dapat menggunakan komponen ini jika Anda memiliki pengaturan jaringan yang kompleks, seperti di mana router meneruskan port broker MQTT ke perangkat inti.

  Anda tidak diharuskan mengonfigurasi komponen ini untuk menggunakannya.

- Pengelola bayangan (`aws.greengrass.ShadowManager`)

  > **Note**
  >
  > Untuk mengelola bayangan perangkat klien, Anda harus menggunakan Greengrass nucleus v2.6.0 atau yang lebih baru, shadow manager v2.2.0 atau yang lebih baru, dan MQTT bridge v2.2.0 atau yang lebih baru.

  (Opsional) Menyebarkan komponen shadow manager untuk mengelola bayangan perangkat klien pada perangkat inti. Komponen Greengrass bisa mendapatkan, memperbarui, dan

menghapus bayangan perangkat klien untuk berinteraksi dengan perangkat klien. Anda juga dapat mengonfigurasi komponen shadow manager untuk menyinkronkan bayangan perangkat klien dengan layanan AWS IoT Core cloud.

Untuk menggunakan komponen ini dengan bayangan perangkat klien, Anda harus mengonfigurasi komponen jembatan MQTT untuk menyampaikan pesan antara perangkat klien dan pengelola bayangan, yang menggunakan penerbitan/langganan lokal. Jika tidak, komponen ini tidak memerlukan konfigurasi untuk digunakan, tetapi memerlukan konfigurasi untuk menyinkronkan bayangan perangkat.

> **Note**
>
> Kami menyarankan Anda menerapkan hanya satu komponen broker MQTT. Jembatan MQTT dan komponen detektor IP bekerja dengan hanya satu komponen broker MQTT pada satu waktu. Jika Anda menggunakan beberapa komponen broker MQTT, Anda harus mengonfigurasinya untuk menggunakan port yang berbeda.

Konfigurasikan penemuan cloud (konsol)

Anda dapat menggunakan konsol AWS IoT Greengrass untuk mengaitkan perangkat klien, mengelola titik akhir perangkat, dan men-deploy komponen untuk memungkinkan perangkat klien. Untuk informasi selengkapnya, lihat Langkah 2: Aktifkan dukungan perangkat klien.

Konfigurasikan penemuan cloud (AWS CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk mengaitkan perangkat klien, mengelola titik akhir perangkat, dan men-deploy komponen untuk memungkinkan dukungan perangkat klien. Untuk informasi selengkapnya, lihat informasi berikut:

- Kelola asosiasi perangkat klien (AWS CLI)
- Kelola titik akhir perangkat inti
- Komponen perangkat klien yang disediakan oleh AWS
- Buat deployment

Kaitkan perangkat klien

Untuk menggunakan penemuan cloud, kaitkan perangkat klien ke perangkat inti agar dapat menemukan perangkat inti. Kemudian, perangkat-perangkat itu dapat menggunakan API Penemuan Greengrass untuk mengambil informasi konektivitas dan sertifikat untuk perangkat inti terkait mereka.

Demikian juga, pisahkan perangkat klien dari perangkat inti untuk menghentikannya dari menemukan perangkat inti.

Topik

- Kelola asosiasi perangkat klien (konsol)
- Kelola asosiasi perangkat klien (AWS CLI)
- Kelola asosiasi perangkat klien (API)

Kelola asosiasi perangkat klien (konsol)

Anda dapat menggunakan konsol AWS IoT Greengrass untuk melihat, menambah, dan menghapus asosiasi perangkat klien.

Untuk melihat asosiasi perangkat klien untuk perangkat inti (konsol)

1. Navigasikan ke konsol AWS IoT Greengrass tersebut.
2. Pilih Perangkat inti.
3. Pilih perangkat inti untuk dikelola.
4. Pada halaman detail perangkat inti, pilih tab Perangkat klien.
5. Di bagian Perangkat klien terkait, Anda dapat melihat perangkat klien mana (objek AWS IoT) yang terkait dengan perangkat inti.

Untuk mengaitkan perangkat klien dengan perangkat inti (konsol)

1. Navigasikan ke konsol AWS IoT Greengrass tersebut.
2. Pilih Perangkat inti.
3. Pilih perangkat inti untuk dikelola.
4. Pada halaman detail perangkat inti, pilih tab Perangkat klien.
5. Di bagian Perangkat klien terkait, pilih Kaitkan perangkat klien.

6. Di modal Kaitkan perangkat klien dengan perangkat inti, lakukan hal berikut untuk setiap perangkat klien yang akan dikaitkan:

   a. Masukkan nama objek AWS IoT yang akan diasosiasikan sebagai perangkat klien.

   b. Pilih Tambahkan.

7. Pilih Kaitkan.

   Perangkat klien yang telah Anda kaitkan sekarang dapat menggunakan Greengrass penemuan API untuk menemukan perangkat inti ini.

Untuk mengaitkan perangkat klien dengan perangkat inti (konsol)

1. Navigasikan ke konsol AWS IoT Greengrass tersebut.
2. Pilih Perangkat inti.
3. Pilih perangkat inti untuk dikelola.
4. Pada halaman detail perangkat inti, pilih tab Perangkat klien.
5. Di bagian Perangkat klien terkait, pilih setiap perangkat klien yang akan dipisahkan.
6. Pilih Pisahkan.
7. Dalam modal konfirmasi, pilih Pisahkan.

   Perangkat klien yang telah Anda pisahkan sekarang dapat menggunakan Greengrass penemuan API untuk menemukan perangkat inti ini.

Kelola asosiasi perangkat klien (AWS CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk mengelola asosiasi perangkat klien untuk perangkat inti.

Untuk melihat asosiasi perangkat klien untuk perangkat inti (AWS CLI)

- Gunakan perintah berikut: list-client-devices-associated- with-core-device.

Untuk mengaitkan perangkat klien dengan perangkat inti (AWS CLI)

- Gunakan perintah berikut: batch-associate-client-device- with-core-device.

Untuk memisahkan perangkat klien dari perangkat inti (AWS CLI)

- Gunakan perintah berikut: batch-disassociate-client-device- from-core-device.

Kelola asosiasi perangkat klien (API)

Anda dapat menggunakan API AWS untuk mengelola asosiasi perangkat klien untuk perangkat inti.

Untuk melihat asosiasi perangkat klien untuk perangkat inti (API AWS)

- Gunakan operasi berikut: ListClientDevicesAssociatedWithCoreDevice.

Untuk mengaitkan perangkat klien dengan perangkat inti (API AWS)

- Gunakan operasi berikut: BatchAssociateClientDeviceWithCoreDevice.

Untuk memisahkan perangkat klien dari perangkat inti (API AWS)

- Gunakan operasi berikut: BatchDisassociateClientDeviceFromCoreDevice.

Mengautentikasi klien saat offline

Dengan otentikasi offline, Anda dapat mengonfigurasi perangkat AWS IoT Greengrass Core sehingga perangkat klien dapat terhubung ke perangkat inti, bahkan ketika perangkat inti tidak terhubung ke cloud. Saat Anda menggunakan otentikasi offline, perangkat Greengrass Anda dapat terus bekerja di lingkungan offline sebagian.

Untuk menggunakan otentikasi offline untuk perangkat klien dengan koneksi ke cloud, Anda memerlukan yang berikut ini:

- Perangkat AWS IoT Greengrass inti dengan Auth perangkat klien komponen yang digunakan. Anda harus menggunakan versi 2.3.0 atau lebih tinggi untuk otentikasi offline.
- Koneksi cloud untuk perangkat inti selama koneksi awal perangkat klien.

Menyimpan kredensi klien

Ketika perangkat klien terhubung ke perangkat inti untuk pertama kalinya, perangkat inti memanggil AWS IoT Greengrass layanan. Saat dipanggil, Greengrass memvalidasi pendaftaran perangkat

klien sebagai sesuatu. AWS IoT Ini juga memvalidasi bahwa perangkat memiliki sertifikat yang valid. Perangkat inti kemudian menyimpan informasi ini secara lokal.

Lain kali perangkat terhubung, perangkat inti Greengrass mencoba memvalidasi perangkat klien dengan layanan. AWS IoT Greengrass Jika tidak dapat terhubung AWS IoT Greengrass, perangkat inti menggunakan informasi perangkat yang disimpan secara lokal untuk memvalidasi perangkat klien.

Anda dapat mengonfigurasi lamanya waktu perangkat inti Greengrass menyimpan kredensional. Anda dapat mengatur batas waktu dari satu menit menjadi 2.147,483.647 menit dengan menyetel opsi `clientDeviceTrustDurationMinutes` konfigurasi dalam konfigurasi komponen autentikasi perangkat klien. Defaultnya adalah satu menit, yang secara efektif mematikan otentikasi offline. Saat Anda menetapkan batas waktu ini, kami sarankan Anda mempertimbangkan kebutuhan keamanan Anda. Anda juga harus mempertimbangkan berapa lama Anda mengharapkan perangkat inti berjalan saat terputus dari cloud.

Perangkat inti memperbarui penyimpanan kredensialnya sebanyak tiga kali:

1. Saat perangkat terhubung ke perangkat inti untuk pertama kalinya.
2. Jika perangkat inti terhubung ke cloud, ketika perangkat klien terhubung kembali ke perangkat inti.
3. Jika perangkat inti terhubung ke cloud, sekali sehari untuk menyegarkan seluruh toko kredensi.

Saat perangkat inti Greengrass menyegarkan toko kredensialnya, ia menggunakan operasi. ListClientDevicesAssociatedWithCoreDevice Greengrass hanya menyegarkan perangkat yang dikembalikan oleh operasi ini. Untuk mengaitkan perangkat klien dengan perangkat inti, lihatKaitkan perangkat klien.

Untuk menggunakan `ListClientDevicesAssociatedWithCoreDevice` operasi, Anda harus menambahkan izin untuk operasi ke peran AWS Identity and Access Management (IAM) yang terkait dengan Akun AWS yang berjalan AWS IoT Greengrass. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Kelola titik akhir perangkat inti

Ketika Anda menggunakan penemuan cloud, Anda menyimpan titik akhir broker MQTT untuk perangkat inti di layanan cloud AWS IoT Greengrass. Perangkat klien tersambung ke AWS IoT Greengrass untuk mengambil titik akhir ini dan informasi lainnya untuk perangkat inti terkaitnya.

Untuk setiap perangkat inti, Anda dapat secara otomatis atau manual mengelola titik akhir.

- Secara otomatis mengelola titik akhir dengan detektor IP

  Anda dapat menerapkan komponen detektor IP untuk secara otomatis mengelola titik akhir perangkat inti untuk Anda jika Anda memiliki pengaturan jaringan yang tidak kompleks, seperti di mana perangkat klien berada di jaringan yang sama dengan perangkat inti. Anda tidak dapat menggunakan komponen detektor IP jika perangkat inti berada di belakang router yang meneruskan port broker MQTT ke perangkat inti, misalnya.

  Komponen detektor IP juga berguna jika Anda men-deploy ke grup objek, karena ia mengelola titik akhir untuk semua perangkat inti dalam grup objek. Untuk informasi selengkapnya, lihat Gunakan detektor IP untuk mengelola titik akhir secara otomatis.
- Kelola titik akhir secara manual

  Jika Anda tidak dapat menggunakan komponen detektor IP, Anda harus mengelola titik akhir perangkat inti secara manual. Anda dapat memperbarui titik akhir ini dengan konsol atau API. Untuk informasi selengkapnya, lihat Kelola titik akhir secara manual.

Topik

Gunakan detektor IP untuk mengelola titik akhir secara otomatis

Jika Anda memiliki pengaturan jaringan sederhana, seperti perangkat klien pada jaringan yang sama sebagai perangkat inti, Anda dapatmen-deploy Komponen pendeteksi IP untuk melakukan hal berikut:

- Memantau informasi konektivitas jaringan lokal perangkat inti Greengrass. Informasi ini mencakup titik akhir jaringan perangkat inti dan port tempat broker MQTT beroperasi.
- Laporkan informasi konektivitas perangkat inti ke layanan cloud AWS IoT Greengrass.

Komponen detektor IP menimpa titik akhir yang Anda tetapkan secara manual.

> **Important**
>
> kebijakan AWS IoT perangkat inti harus mengizinkan izin `greengrass:UpdateConnectivityInfo` untuk menggunakan komponen detektor IP.

Lihat informasi yang lebih lengkap di Kebijakan AWS IoT untuk operasi bidang data dan Konfigurasikan kebijakan AWS IoT hal.

Anda dapat melakukan salah satu dari berikut ini untuk men-deploy komponen detektor IP:

- Gunakan halaman Konfigurasi penemuan di konsol. Untuk informasi selengkapnya, lihat Konfigurasikan penemuan cloud (konsol).
- Membuat dan merevisi deployment untuk menyertakan detektor IP. Anda bisa menggunakan konsol, AWS CLI, atau API AWS untuk mengelola deployment. Untuk informasi selengkapnya, lihat Buat deployment.

Menyebarkan komponen detektor IP (konsol)

1. Pada menu navigasi konsol AWS IoT Greengrass tersebut, pilih Komponen.
2. Pada halaman Components, pilih tab Public components, lalu pilih aws.greengrass.clientdevices.IPDetector.
3. Pada halaman aws.greengrass.clientdevices.IPDetector pilih Deploy.
4. Dari Tambahkan ke penerapan, pilih penerapan yang ada untuk direvisi, atau pilih untuk membuat penerapan baru, lalu pilih Berikutnya.
5. Jika Anda memilih untuk membuat penerapan baru, pilih perangkat inti target atau grup hal untuk penerapan. Pada halaman Tentukan target, di bawah target Deployment, pilih perangkat inti atau grup benda, lalu pilih Berikutnya.
6. Pada halaman Pilih komponen, verifikasi bahwa aws.greengrass.clientdevices.IPDetectorkomponen dipilih, pilih Berikutnya.
7. Pada halaman Configure components, pilih aws.greengrass.clientdevices.IPDetector, lalu lakukan hal berikut:

    a. Pilih Konfigurasi komponen.

    b. Dalam aws.greengrass.clientdevices.IPDetector modal Konfigurasi, di bawah pembaruan Konfigurasi, di Konfigurasi untuk digabungkan, Anda dapat memasukkan pembaruan konfigurasi untuk mengonfigurasi komponen detektor IP. Anda dapat menentukan salah satu opsi konfigurasi berikut:

- `defaultPort`— (Opsional) Port broker MQTT untuk melaporkan kapan komponen ini mendeteksi alamat IP. Anda harus menentukan parameter ini jika Anda mengonfigurasi broker MQTT untuk menggunakan port yang berbeda dari port default 8883.
- `includeIPv4LoopbackAddrs`— (Opsional) Anda dapat mengaktifkan opsi ini untuk mendeteksi dan melaporkan alamat loopback IPv4. Ini adalah alamat IP, seperti`localhost`, di mana perangkat dapat berkomunikasi dengan dirinya sendiri. Gunakan opsi ini di lingkungan pengujian tempat perangkat inti dan perangkat klien berjalan pada sistem yang sama.
- `includeIPv4LinkLocalAddrs`— (Opsional) Anda dapat mengaktifkan opsi ini untuk mendeteksi dan melaporkan alamat link-lokal IPv4. Gunakan opsi ini jika jaringan perangkat inti tidak memiliki Dynamic Host Configuration Protocol (DHCP) atau alamat IP yang ditetapkan secara statis.

Pembaruan konfigurasi mungkin terlihat mirip dengan contoh berikut.

```
{
  "defaultPort": "8883",
  "includeIPv4LoopbackAddrs": false,
  "includeIPv4LinkLocalAddrs": false
}
```

c. Pilih Konfirmasi untuk menutup modal, lalu pilih Berikutnya.

8. Pada halaman Konfigurasikan pengaturan lanjutan, simpan pengaturan konfigurasi default tersebut, dan pilih Selanjutnya.
9. Di halaman Tinjau, pilih Deploy.

   Penyebaran dapat memakan waktu hingga satu menit untuk diselesaikan.

Menyebarkan komponen detektor IP () AWS CLI

Untuk menyebarkan komponen detektor IP, buat dokumen penerapan yang disertakan `aws.greengrass.clientdevices.IPDetector` dalam `components` objek, dan tentukan pembaruan konfigurasi untuk komponen tersebut. Ikuti petunjuk Buat deployment untuk membuat penerapan baru atau merevisi penerapan yang ada.

Anda dapat menentukan salah satu opsi berikut untuk mengonfigurasi komponen detektor IP saat Anda membuat dokumen penerapan:

- `defaultPort`— (Opsional) Port broker MQTT untuk melaporkan kapan komponen ini mendeteksi alamat IP. Anda harus menentukan parameter ini jika Anda mengonfigurasi broker MQTT untuk menggunakan port yang berbeda dari port default 8883.
- `includeIPv4LoopbackAddrs`— (Opsional) Anda dapat mengaktifkan opsi ini untuk mendeteksi dan melaporkan alamat loopback IPv4. Ini adalah alamat IP, seperti`localhost`, di mana perangkat dapat berkomunikasi dengan dirinya sendiri. Gunakan opsi ini di lingkungan pengujian tempat perangkat inti dan perangkat klien berjalan pada sistem yang sama.
- `includeIPv4LinkLocalAddrs`— (Opsional) Anda dapat mengaktifkan opsi ini untuk mendeteksi dan melaporkan alamat link-lokal IPv4. Gunakan opsi ini jika jaringan perangkat inti tidak memiliki Dynamic Host Configuration Protocol (DHCP) atau alamat IP yang ditetapkan secara statis.

Contoh berikut dokumen penyebaran sebagian menentukan untuk melaporkan port 8883 sebagai port broker MQTT.

```
{
  ...,
  "components": {
    ...,
    "aws.greengrass.clientdevices.IPDetector": {
      "componentVersion": "2.1.1",
      "configurationUpdate": {
        "merge": "{\"defaultPort\":\"8883\",}"
      }
    }
  }
}
```

Kelola titik akhir secara manual

Anda dapat secara otomatis mengelola titik akhir broker MQTT untuk perangkat inti.

Setiap titik akhir broker MQTT memiliki informasi berikut:

Titik akhir () `HostAddress`

Alamat IP atau alamat DNS di mana perangkat klien dapat terhubung ke broker MQTT pada perangkat inti.

Port (`PortNumber`)

Port tempat broker MQTT beroperasi pada perangkat inti.

Anda dapat mengonfigurasi port ini pada Komponen broker MQTT Moquette, yang defaultnya menggunakan port 8883.

Metadata () `Metadata`

Metadata tambahan yang akan diberikan ke perangkat klien yang terhubung ke titik akhir ini.

Topik

- Mengelola titik akhir (konsol)
- Kelola titik akhir (AWS CLI)
- Kelola titik akhir (API)

Mengelola titik akhir (konsol)

Anda dapat menggunakan konsol AWS IoT Greengrass untuk melihat, memperbarui, dan menghapus titik akhir untuk perangkat inti.

Untuk mengelola titik akhir untuk perangkat inti (konsol)

1. Navigasikan ke konsol AWS IoT Greengrass tersebut.
2. Pilih Perangkat inti.
3. Pilih perangkat inti untuk dikelola.
4. Pada halaman detail perangkat inti, pilih tab Perangkat klien.
5. Di Titik akhir broker MQTT, Anda dapat melihat titik akhir broker MQTT perangkat inti. Pilih Kelola titik akhir.
6. Di modal Kelola titik akhir, tambahkan atau hapus titik akhir broker MQTT untuk perangkat inti.
7. Pilih Perbarui.

Kelola titik akhir (AWS CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk mengelola titik akhir untuk perangkat inti.

> **Note**
>
> Karena dukungan perangkat klien di AWS IoT Greengrass V2 kompatibel dengan versi mundurAWS IoT Greengrass V1, Anda dapat menggunakan AWS IoT Greengrass V2 atau operasi AWS IoT Greengrass V1 API untuk mengelola titik akhir perangkat inti.

Untuk mendapatkan titik akhir untuk perangkat inti (AWS CLI)

- Gunakan salah satu dari perintah berikut:
  - greengrassv2: get-connectivity-info
  - greengrass: get-connectivity-info

Untuk memperbarui titik akhir untuk perangkat inti (AWS CLI)

- Gunakan salah satu dari perintah berikut:
  - greengrassv2: update-connectivity-info
  - greengrass: update-connectivity-info

Kelola titik akhir (API)

Anda dapat menggunakan API AWS untuk mengelola titik akhir untuk perangkat inti.

> **Note**
>
> Karena dukungan perangkat klien di AWS IoT Greengrass V2 kompatibel dengan versi mundurAWS IoT Greengrass V1, Anda dapat menggunakan AWS IoT Greengrass V2 atau operasi AWS IoT Greengrass V1 API untuk mengelola titik akhir perangkat inti.

Untuk mendapatkan titik akhir untuk perangkat inti (API AWS)

- Gunakan salah satu dari operasi berikut:
  - V2: GetConnectivityInfo
  - V1: GetConnectivityInfo

Untuk memperbarui titik akhir untuk perangkat inti (API AWS)

- Gunakan salah satu dari operasi berikut:
  - V2: UpdateConnectivityInfo
  - V1: UpdateConnectivityInfo

Pilih broker MQTT

AWS IoT Greengrassmenyediakan opsi bagi Anda untuk memilih broker MQTT lokal mana yang akan dijalankan di perangkat inti Anda. Perangkat klien terhubung ke broker MQTT yang berjalan pada perangkat inti, jadi pilihlah broker MQTT yang kompatibel dengan perangkat klien yang ingin Anda sambungkan.

> **Note**
>
> Kami menyarankan Anda menerapkan hanya satu komponen broker MQTT. Jembatan MQTT dan komponen detektor IP bekerja dengan hanya satu komponen broker MQTT pada satu waktu. Jika Anda menggunakan beberapa komponen broker MQTT, Anda harus mengonfigurasinya untuk menggunakan port yang berbeda.

Anda dapat memilih dari broker MQTT berikut:

- MQTT 3.1.1 broker (Moquette) — `aws.greengrass.clientdevices.mqtt.Moquette`

  Pilih opsi ini untuk broker MQTT ringan yang sesuai dengan standar MQTT 3.1.1. Broker AWS IoT Core MQTT dan AWS IoT Device SDK juga sesuai dengan standar MQTT 3.1.1, sehingga Anda dapat menggunakan fitur ini untuk membuat aplikasi yang menggunakan MQTT 3.1.1 di seluruh perangkat Anda dan. AWS Cloud
- Pialang MQTT 5 (EMQX) - `aws.greengrass.clientdevices.mqtt.EMQX`

  Pilih opsi ini untuk menggunakan fitur MQTT 5 dalam komunikasi antara perangkat inti dan perangkat klien. Komponen ini menggunakan lebih banyak sumber daya daripada broker Moquette MQTT 3.1.1, dan pada perangkat inti Linux, ia membutuhkan Docker.

  MQTT 5 kompatibel dengan MQTT 3.1.1, sehingga Anda dapat menghubungkan perangkat klien yang menggunakan MQTT 3.1.1 ke broker ini. Jika Anda menjalankan broker Moquette MQTT

3.1.1, Anda dapat menggantinya dengan broker EMQX MQTT 5, dan perangkat klien dapat terus terhubung dan beroperasi seperti biasa.

- Menerapkan broker khusus

  Pilih opsi ini untuk membuat komponen broker lokal khusus untuk berkomunikasi dengan perangkat klien. Anda dapat membuat broker lokal khusus yang menggunakan protokol selain MQTT. AWS IoT Greengrassmenyediakan SDK komponen yang dapat Anda gunakan untuk mengautentikasi dan mengotorisasi perangkat klien. Lihat informasi yang lebih lengkap di Gunakan AWS IoT Device SDK untuk berkomunikasi dengan inti Greengrass, komponen lain, dan AWS IoT Core dan Otentikasi dan otorisasi perangkat klien.

Menghubungkan perangkat klien ke perangkatAWS IoT Greengrass Core dengan broker MQTT

Saat Anda menggunakan broker MQTT di perangkatAWS IoT Greengrass Core Anda, perangkat menggunakan otoritas sertifikat perangkat inti (CA) yang unik untuk perangkat untuk mengeluarkan sertifikat kepada broker untuk membuat koneksi TLS bersama dengan klien.

AWS IoT Greengrassakan membuat perangkat inti CA, atau Anda dapat memberikan milik Anda sendiri. CA perangkat inti terdaftar denganAWS IoT Greengrass ketikaAuth perangkat klien komponen terhubung. CA perangkat inti yang dibuat otomatis bersifat persisten, perangkat akan terus menggunakan CA yang sama selama komponen autentikasi perangkat klien dikonfigurasi.

Ketika broker MQTT dimulai, ia meminta sertifikat. Komponen autentikasi perangkat klien mengeluarkan sertifikat X.509 menggunakan CA perangkat inti. Sertifikat diputar ketika broker mulai, ketika sertifikat kedaluwarsa, atau ketika informasi konektivitas seperti alamat IP berubah. Untuk informasi selengkapnya, lihat Rotasi sertifikat pada broker MQTT lokal.

Untuk menghubungkan klien ke broker MQTT, Anda memerlukan hal berikut ini:

- Perangkat klien harus memiliki perangkatAWS IoT Greengrass Core CA. Anda bisa mendapatkan CA ini melalui cloud discovery, atau dengan menyediakan CA secara manual. Untuk informasi selengkapnya, lihat Menggunakan otoritas sertifikat Anda sendiri.
- Nama domain yang memenuhi syarat (FQDN) atau alamat IP dari perangkat inti harus ada di sertifikat broker yang dikeluarkan oleh CA perangkat inti. Anda memastikan ini menggunakanDetektor IP komponen atau mengkonfigurasi alamat IP secara manual. Untuk informasi selengkapnya, lihat Kelola titik akhir perangkat inti.

- Komponen auth perangkat klien harus memberikan izin perangkat klien untuk terhubung ke perangkat core Greengrass. Untuk informasi selengkapnya, lihat Auth perangkat klien.

Menggunakan otoritas sertifikat Anda sendiri

Jika perangkat klien Anda tidak dapat mengakses cloud untuk menemukan perangkat inti Anda, Anda dapat memberikan otoritas sertifikat perangkat inti (CA). Perangkat inti Greengrass Anda menggunakan perangkat inti CA untuk menerbitkan sertifikat untuk broker MQTT Anda. Setelah Anda mengonfigurasi perangkat inti dan menyediakan perangkat klien Anda dengan CA, perangkat klien Anda dapat terhubung ke titik akhir dan memverifikasi jabat tangan TLS menggunakan perangkat inti CA (CA yang disediakan sendiri atau dibuat secara otomatis).

Untuk mengonfigurasiAuth perangkat klien komponen agar menggunakan CA perangkat inti Anda, atur parameter`certificateAuthority` konfigurasi saat Anda menerapkan komponen. Selama konfigurasi, Anda harus memberikan rincian berikut:

- Lokasi dari sertifikat CA perangkat inti.
- Kunci privat dari sertifikat CA perangkat inti.
- (Opsional) Rantai sertifikat ke sertifikat root jika perangkat inti CA adalah CA perantara.

Jika Anda menyediakan CA perangkat inti,AWS IoT Greengrass daftarkan CA dengan cloud.

Anda dapat menyimpan sertifikat Anda dalam modul keamanan perangkat keras atau pada sistem file. Contoh berikut menunjukkan`certificateAuthority` konfigurasi untuk CA menengah disimpan menggunakan HSM/TPM. Perhatikan bahwa rantai sertifikat hanya dapat disimpan pada disk.

```
"certificateAuthority": {
    "certificateUri": "pkcs11:object=CustomerIntermediateCA;type=cert",
    "privateKeyUri": "pkcs11:object=CustomerIntermediateCA;type=private"
    "certificateChainUri": "file:///home/ec2-user/creds/certificateChain.pem",
  }
```

Dalam contoh ini, parameter`certificateAuthority` konfigurasi mengkonfigurasi komponen autentikasi perangkat klien untuk menggunakan CA perantara dari sistem file:

```
"certificateAuthority": {
    "certificateUri": "file:///home/ec2-user/creds/intermediateCA.pem",
```

```
    "privateKeyUri": "file:///home/ec2-user/creds/intermediateCA.privateKey.pem",
    "certificateChainUri": "file:///home/ec2-user/creds/certificateChain.pem",
  }
```

Untuk menghubungkan perangkat ke perangkatAWS IoT Greengrass Core Anda, lakukan hal berikut:

1. Buat otoritas sertifikat (CA) untuk perangkat core Greengrass menggunakan CA root organisasi Anda. Kami menyarankan Anda menggunakan CA perantara sebagai praktik terbaik keamanan.
2. Berikan sertifikat CA menengah, kunci privat, dan rantai sertifikat ke CA root Anda ke perangkat core Greengrass. Untuk informasi selengkapnya, lihat Auth perangkat klien. CA perantara menjadi CA perangkat inti untuk perangkat inti Greengrass, dan perangkat mendaftarkan CA denganAWS IoT Greengrass.
3. Daftarkan perangkat klien sebagaiAWS IoT sesuatu. Untuk informasi selengkapnya, lihat Membuat objek benda di PanduanAWS IoT Core Pengembang. Tambahkan kunci pribadi, kunci publik, sertifikat perangkat, dan sertifikat CA root ke perangkat klien Anda. Bagaimana Anda menambahkan informasi tergantung pada perangkat dan perangkat lunak Anda.

Setelah Anda mengonfigurasi perangkat, Anda dapat menggunakan sertifikat dan gantungan kunci publik untuk terhubung ke perangkat core Greengrass. Perangkat lunak Anda bertanggung jawab untuk menemukan titik akhir perangkat inti. Anda dapat mengatur titik akhir secara manual untuk perangkat inti. Untuk informasi selengkapnya, lihat Kelola titik akhir secara manual.

Uji komunikasi perangkat klien

Perangkat klien dapat menggunakan AWS IoT Device SDK untuk menemukan, menghubungkan, dan berkomunikasi dengan perangkat inti. Anda dapat menggunakan klien penemuan Greengrass di AWS IoT Device SDK untuk menggunakan API Penemuan Greengrass, yang mengembalikan informasi tentang perangkat inti yang dapat dihubungkan oleh perangkat klien. Respon API mencakup titik akhir broker MQTT untuk untuk terhubung dan sertifikat untuk digunakan untuk memverifikasi identitas setiap perangkat inti. Kemudian, perangkat klien dapat mencoba setiap titik akhir sampai berhasil terhubung ke perangkat inti.

Perangkat klien hanya dapat menemukan perangkat inti yang Anda kaitkan. Sebelum Anda menguji komunikasi antara perangkat klien dan perangkat inti, Anda harus mengaitkan perangkat klien ke perangkat inti. Untuk informasi selengkapnya, lihat Kaitkan perangkat klien.

API penemuan Greengrass mengembalikan titik akhir broker perangkat inti MQTT yang Anda tentukan. Anda dapat menggunakan Komponen pendeteksi IP untuk mengelola titik akhir ini untuk

Anda, atau Anda dapat mengelolanya secara manual untuk setiap perangkat inti. Untuk informasi selengkapnya, lihat Kelola titik akhir perangkat inti.

> **Note**
>
> Untuk menggunakan API penemuan Greengrass, perangkat klien harus memiliki izin `greengrass:Discover`. Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat klien.

AWS IoT Device SDK tersedia dalam beragam bahasa pemrograman. Untuk informasi selengkapnya, lihat SDK Perangkat AWS IoT di Panduan Developer AWS IoT Core.

Topik

- Uji komunikasi (Python)
- Uji komunikasi (C++)
- Uji komunikasi (JavaScript)
- Uji komunikasi (Java)

Uji komunikasi (Python)

Pada bagian ini, Anda menggunakan sampel penemuan Greengrass di v2 for Python AWS IoT Device SDK untuk menguji komunikasi antara perangkat klien dan perangkat inti.

> **Important**
>
> Untuk menggunakan v2 for Python AWS IoT Device SDK, perangkat harus menjalankan Python 3.6 atau yang lebih baru.

Untuk menguji komunikasi (v2 for Python AWS IoT Device SDK)

1. Unduh dan instal v2 for Python AWS IoT Device SDK ke objek AWS IoT untuk terhubung sebagai perangkat klien.

   Di perangkat klien, lakukan hal berikut:

   a. Kloning v2 for Python repository AWS IoT Device SDK untuk mengunduhnya.

```
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

b. Instal v2 for Python AWS IoT Device SDK.

```
python3 -m pip install --user ./aws-iot-device-sdk-python-v2
```

2. Ubah ke folder contoh di v2 for Python AWS IoT Device SDK.

```
cd aws-iot-device-sdk-python-v2/samples
```

3. Jalankan sampel aplikasi penemuan Greengrass. Aplikasi ini mengharapkan argumen yang menentukan nama objek perangkat klien, topik MQTT dan pesan yang akan digunakan, dan sertifikat yang mengautentikasi dan mengamankan sambungan. Contoh berikut mengirimkan pesan Hello World ke topik `clients/`*`MyClientDevice1`*`/hello/world`.

   - Ganti *`MyClientDevice1`* dengan nama benda perangkat klien.
   - Ganti *`~/certs/ AmazonRoot ca1.pem dengan`* jalur ke sertifikat CA root Amazon di perangkat klien.
   - Ganti *`~/certs/device.pem.crt`* dengan jalur ke sertifikat perangkat pada perangkat klien.
   - Ganti *`~/certs/private.pem.key`* dengan jalur menuju file kunci pribadi pada perangkat klien.
   - Ganti *`us-east-1`* dengan Wilayah AWS di mana perangkat klien dan perangkat inti Anda beroperasi.

```
python3 basic_discovery.py \\
  --thing_name MyClientDevice1 \\
  --topic 'clients/MyClientDevice1/hello/world' \\
  --message 'Hello World!' \\
  --ca_file ~/certs/AmazonRootCA1.pem \\
  --cert ~/certs/device.pem.crt \\
  --key ~/certs/private.pem.key \\
  --region us-east-1 \\
  --verbosity Warn
```

Aplikasi sampel penemuan mengirimkan pesan 10 kali dan terputus. Aplikasi ini juga berlangganan topik yang sama di mana ia menerbitkan pesan. Jika output menunjukkan bahwa

aplikasi itu menerima pesan MQTT pada topik, perangkat klien dapat berhasil berkomunikasi dengan perangkat inti.

```
Performing greengrass discovery...
awsiot.greengrass_discovery.DiscoverResponse(gg_groups=[awsiot.greengrass_discovery.GGGroup
coreDevice-MyGreengrassCore',
 cores=[awsiot.greengrass_discovery.GGCore(thing_arn='arn:aws:iot:us-
east-1:123456789012:thing/MyGreengrassCore',
 connectivity=[awsiot.greengrass_discovery.ConnectivityInfo(id='203.0.113.0',
 host_address='203.0.113.0', metadata='', port=8883)])],
 certificate_authorities=['-----BEGIN CERTIFICATE-----\
MIICiT...EXAMPLE=\
-----END CERTIFICATE-----\
'])])
Trying core arn:aws:iot:us-east-1:123456789012:thing/MyGreengrassCore at host
 203.0.113.0 port 8883
Connected!
Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 0}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 0}'
Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 1}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 1}'

...

Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 9}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 9}'
```

Jika aplikasi mengeluarkan kesalahan, lihat Pemecahan masalah penemuan Greengrass.

Anda juga dapat melihat log Greengrass pada perangkat inti untuk memverifikasi apakah perangkat klien berhasil menghubungkan dan mengirim pesan. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Uji komunikasi (C++)

Pada bagian ini, Anda menggunakan sampel penemuan Greengrass di v2 for C++ AWS IoT Device SDK untuk menguji komunikasi antara perangkat klien dan perangkat inti.

Untuk membangun v2 for C++ AWS IoT Device SDK, perangkat harus mempunyai alat berikut:

- C++ 11 atau yang lebih baru
- CMake 3.1 atau yang lebih baru
- Salah satu penyusun berikut:
  - GCC 4.8 atau yang lebih baru
  - Clang 3.9 atau yang lebih baru
  - MSVC 2015 atau yang lebih baru

Untuk menguji komunikasi (v2 for C++ AWS IoT Device SDK)

1. Unduh dan bangun v2 for C++ AWS IoT Device SDK ke objek AWS IoT untuk terhubung sebagai perangkat klien.

   Di perangkat klien, lakukan hal berikut:

   a. Buat folder untuk workspace v2 for C++ AWS IoT Device SDK, dan ubah ke itu.

   ```
   cd
   mkdir iot-device-sdk-cpp
   cd iot-device-sdk-cpp
   ```

   b. Kloning v2 for C++ repository AWS IoT Device SDK untuk mengunduhnya. Bendera `--recursive` menentukan untuk men-download submodul.

   ```
   git clone --recursive https://github.com/aws/aws-iot-device-sdk-cpp-v2.git
   ```

   c. Buat folder untuk output bangunan v2 for C++ AWS IoT Device SDK, dan ubah ke itu.

   ```
   mkdir aws-iot-device-sdk-cpp-v2-build
   cd aws-iot-device-sdk-cpp-v2-build
   ```

   d. Bangun v2 for C++ AWS IoT Device SDK.

```
cmake -DCMAKE_INSTALL_PREFIX="~/iot-device-sdk-cpp" -
DCMAKE_BUILD_TYPE="Release" ../aws-iot-device-sdk-cpp-v2
cmake --build . --target install
```

2. Bangun aplikasi sampel penemuan Greengrass di v2 for C++ AWS IoT Device SDK. Lakukan hal-hal berikut:

   a. Ubah ke folder sampel penemuan Greengrass di v2 for C++ AWS IoT Device SDK.

   ```
   cd ../aws-iot-device-sdk-cpp-v2/samples/greengrass/basic_discovery
   ```

   b. Buat folder untuk output build sampel penemuan Greengrass, dan ubah ke itu.

   ```
   mkdir build
   cd build
   ```

   c. Bangun sampel aplikasi penemuan Greengrass.

   ```
   cmake -DCMAKE_PREFIX_PATH="~/iot-device-sdk-cpp" -
   DCMAKE_BUILD_TYPE="Release" ..
   cmake --build . --config "Release"
   ```

3. Jalankan sampel aplikasi penemuan Greengrass. Aplikasi ini mengharapkan argumen yang menentukan nama objek perangkat klien, topik MQTT dan pesan yang akan digunakan, dan sertifikat yang mengautentikasi dan mengamankan sambungan. Contoh berikut berlangganan topik `clients/`*`MyClientDevice1`*`/hello/world` dan menerbitkan pesan yang Anda masukkan pada baris perintah untuk topik yang sama.

   - Ganti *`MyClientDevice1`* dengan nama benda perangkat klien.
   - Ganti *`~/certs/ AmazonRoot ca1.pem dengan`* jalur ke sertifikat CA root Amazon di perangkat klien.
   - Ganti *`~/certs/device.pem.crt`* dengan jalur ke sertifikat perangkat pada perangkat klien.
   - Ganti *`~/certs/private.pem.key`* dengan jalur menuju file kunci pribadi pada perangkat klien.
   - Ganti *`us-east-1`* dengan Wilayah AWS di mana perangkat klien dan perangkat inti Anda beroperasi.

```
./basic-discovery \
  --thing_name MyClientDevice1 \
  --topic 'clients/MyClientDevice1/hello/world' \
  --ca_file ~/certs/AmazonRootCA1.pem \
  --cert ~/certs/device.pem.crt \
  --key ~/certs/private.pem.key \
  --region us-east-1
```

Aplikasi sampel penemuan berlangganan topik dan meminta Anda untuk memasukkan pesan untuk dipublikasikan.

```
Connecting to group greengrassV2-coreDevice-MyGreengrassCore with thing arn
 arn:aws:iot:us-east-1:123456789012:thing/MyGreengrassCore, using endpoint
 203.0.113.0:8883
Connected to group greengrassV2-coreDevice-MyGreengrassCore, using connection to
 203.0.113.0:8883
Successfully subscribed to clients/MyClientDevice1/hello/world
Enter the message you want to publish to topic clients/MyClientDevice1/hello/world
 and press enter. Enter 'exit' to exit this program.
```

Jika aplikasi mengeluarkan kesalahan, lihat Pemecahan masalah penemuan Greengrass.

4. Masukkan pesan, seperti **`Hello World!`**.

```
Enter the message you want to publish to topic clients/MyClientDevice1/hello/world
 and press enter. Enter 'exit' to exit this program.
Hello World!
```

Jika output menunjukkan bahwa aplikasi itu menerima pesan MQTT pada topik, perangkat klien dapat berhasil berkomunikasi dengan perangkat inti.

```
Operation on packetId 2 Succeeded
Publish received on topic clients/MyClientDevice1/hello/world
Message:
Hello World!
```

Anda juga dapat melihat log Greengrass pada perangkat inti untuk memverifikasi apakah perangkat klien berhasil menghubungkan dan mengirim pesan. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Uji komunikasi (JavaScript)

Di bagian ini, Anda menggunakan sampel penemuan Greengrass di AWS IoT Device SDKv2 JavaScript untuk menguji komunikasi antara perangkat klien dan perangkat inti.

> **Important**
>
> Untuk menggunakan AWS IoT Device SDK v2 for JavaScript, perangkat harus menjalankan Node v10.0 atau yang lebih baru.

Untuk menguji komunikasi (AWS IoT Device SDKv2 untuk JavaScript)

1. Unduh dan instal AWS IoT Device SDKv2 JavaScript untuk AWS IoT hal yang akan dihubungkan sebagai perangkat klien.

   Di perangkat klien, lakukan hal berikut:

   a. Kloning AWS IoT Device SDK v2 untuk JavaScript repositori untuk mengunduhnya.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
   ```

   b. Instal AWS IoT Device SDK v2 untuk JavaScript.

   ```
   cd aws-iot-device-sdk-js-v2
   npm install
   ```

2. Ubah ke folder sampel penemuan Greengrass di v2 untuk. AWS IoT Device SDK JavaScript

   ```
   cd samples/node/basic_discovery
   ```

3. Instal sampel aplikasi penemuan Greengrass.

   ```
   npm install
   ```

4. Jalankan sampel aplikasi penemuan Greengrass. Aplikasi ini mengharapkan argumen yang menentukan nama objek perangkat klien, topik MQTT dan pesan yang akan digunakan, dan sertifikat yang mengautentikasi dan mengamankan sambungan. Contoh berikut mengirimkan pesan Hello World ke topik `clients/`*`MyClientDevice1`*`/hello/world`.

   - Ganti *`MyClientDevice1`* dengan nama benda perangkat klien.

- Ganti *~/certs/ AmazonRoot ca1.pem dengan* jalur ke sertifikat CA root Amazon di perangkat klien.
- Ganti *~/certs/device.pem.crt* dengan jalur ke sertifikat perangkat pada perangkat klien.
- Ganti *~/certs/private.pem.key* dengan jalur menuju file kunci pribadi pada perangkat klien.
- Ganti *us-east-1* dengan Wilayah AWS di mana perangkat klien dan perangkat inti Anda beroperasi.

```
node dist/index.js \
  --thing_name MyClientDevice1 \
  --topic 'clients/MyClientDevice1/hello/world' \
  --message 'Hello World!' \
  --ca_file ~/certs/AmazonRootCA1.pem \
  --cert ~/certs/device.pem.crt \
  --key ~/certs/private.pem.key \
  --region us-east-1 \
  --verbose warn
```

Aplikasi sampel penemuan mengirimkan pesan 10 kali dan terputus. Aplikasi ini juga berlangganan topik yang sama di mana ia menerbitkan pesan. Jika output menunjukkan bahwa aplikasi itu menerima pesan MQTT pada topik, perangkat klien dapat berhasil berkomunikasi dengan perangkat inti.

```
Discovery Response:
{"gg_groups":[{"gg_group_id":"greengrassV2-coreDevice-
MyGreengrassCore","cores":[{"thing_arn":"arn:aws:iot:us-
east-1:123456789012:thing/MyGreengrassCore","connectivity":
[{"id":"203.0.113.0","host_address":"203.0.113.0","port":8883,"metadata":""}]}],"certificat
["-----BEGIN CERTIFICATE-----\nMIICiT...EXAMPLE=\n-----END CERTIFICATE-----\n"]}]}
Trying
 endpoint={"id":"203.0.113.0","host_address":"203.0.113.0","port":8883,"metadata":""}
[WARN] [2021-06-12T00:46:45Z] [00007f90c0e8d700] [socket] - id=0x7f90b8018710
 fd=26: setsockopt() for NO_SIGNAL failed with errno 92. If you are having SIGPIPE
 signals thrown, you may want to install a signal trap in your application layer.
Connected to
 endpoint={"id":"203.0.113.0","host_address":"203.0.113.0","port":8883,"metadata":""}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":1}
```

```
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":2}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":3}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":4}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":5}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":6}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":7}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":8}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":9}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":10}
Complete!
```

Jika aplikasi mengeluarkan kesalahan, lihat Pemecahan masalah penemuan Greengrass.

Anda juga dapat melihat log Greengrass pada perangkat inti untuk memverifikasi apakah perangkat klien berhasil menghubungkan dan mengirim pesan. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

Uji komunikasi (Java)

Pada bagian ini, Anda menggunakan sampel penemuan Greengrass di v2 for Java AWS IoT Device SDK untuk menguji komunikasi antara perangkat klien dan perangkat inti.

> **Important**
>
> Untuk membangun v2 for Java AWS IoT Device SDK, perangkat harus mempunyai alat berikut:
>
> - Java 8 atau yang lebih baru, dengan `JAVA_HOME` menunjuk ke folder Java.
> - Apache Maven

Untuk menguji komunikasi (v2 for Java AWS IoT Device SDK)

1. Unduh dan instal v2 for Java AWS IoT Device SDK ke objek AWS IoT untuk terhubung sebagai perangkat klien.

   Di perangkat klien, lakukan hal berikut:

   a. Kloning v2 for Java repository AWS IoT Device SDK untuk mengunduhnya.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-java-v2.git
   ```

   b. Ubah ke folder v2 for Java AWS IoT Device SDK.

   c. Bangun v2 for Java AWS IoT Device SDK.

   ```
   cd aws-iot-device-sdk-java-v2
   mvn versions:use-latest-versions -Dincludes="software.amazon.awssdk.crt*"
   mvn clean install
   ```

2. Jalankan sampel aplikasi penemuan Greengrass. Aplikasi ini mengharapkan argumen yang menentukan nama objek perangkat klien, topik MQTT dan pesan yang akan digunakan, dan sertifikat yang mengautentikasi dan mengamankan sambungan. Contoh berikut berlangganan topik `clients/MyClientDevice1/hello/world` dan menerbitkan pesan yang Anda masukkan pada baris perintah untuk topik yang sama.

   - Ganti kedua instance *`MyClientDevice1`* dengan nama benda perangkat klien.
   - Ganti *`$HOME/certs/ AmazonRoot ca1.pem dengan`* jalur ke sertifikat CA root Amazon di perangkat klien.
   - Ganti *`$HOME/certs/device.pem.crt`* dengan jalur ke sertifikat perangkat pada perangkat klien.

- Ganti *$HOME/certs/private.pem.key* dengan jalur menuju file kunci privat pada perangkat klien.
- Ganti *us-east-1* dengan Wilayah AWS di mana perangkat klien Anda dan perangkat inti beroperasi.

```
DISCOVERY_SAMPLE_ARGS="--thing_name MyClientDevice1 \
  --topic 'clients/MyClientDevice1/hello/world' \
  --ca_file $HOME/certs/AmazonRootCA1.pem \
  --cert $HOME/certs/device.pem.crt \
  --key $HOME/certs/private.pem.key \
  --region us-east-1"

mvn exec:java -pl samples/Greengrass \
  -Dexec.mainClass=greengrass.BasicDiscovery \
  -Dexec.args="$DISCOVERY_SAMPLE_ARGS"
```

Aplikasi sampel penemuan berlangganan topik dan meminta Anda untuk memasukkan pesan untuk dipublikasikan.

```
Connecting to group ID greengrassV2-coreDevice-MyGreengrassCore, with thing
 arn arn:aws:iot:us-east-1:123456789012:thing/MyGreengrassCore, using endpoint
 203.0.113.0:8883
Started a clean session
Enter the message you want to publish to topic clients/MyClientDevice1/hello/world
 and press Enter. Type 'exit' or 'quit' to exit this program:
```

Jika aplikasi mengeluarkan kesalahan, lihat Pemecahan masalah penemuan Greengrass.

3. Masukkan pesan, seperti **`Hello World!`**.

```
Enter the message you want to publish to topic clients/MyClientDevice1/hello/world
 and press Enter. Type 'exit' or 'quit' to exit this program:
Hello World!
```

Jika output menunjukkan bahwa aplikasi itu menerima pesan MQTT pada topik, perangkat klien dapat berhasil berkomunikasi dengan perangkat inti.

```
Message received on topic clients/MyClientDevice1/hello/world: Hello World!
```

Anda juga dapat melihat log Greengrass pada perangkat inti untuk memverifikasi apakah perangkat klien berhasil menghubungkan dan mengirim pesan. Lihat informasi yang lebih lengkap di Memantau AWS IoT Greengrass log.

Greengrass discovery RESTful API

AWS IoT Greengrass menyediakan operasi API `Discover` yang dapat digunakan oleh perangkat klien untuk mengidentifikasi perangkat inti Greengrass di mana mereka dapat terhubung. Perangkat klien menggunakan operasi bidang data ini untuk mengambil informasi yang diperlukan untuk terhubung ke perangkat inti Greengrass tempat Anda mengaitkannya dengan operasi API. BatchAssociateClientDeviceWithCoreDevice Ketika perangkat klien online, perangkat dapat terhubung ke layanan cloud AWS IoT Greengrass dan menggunakan API penemuan untuk menemukan:

- Alamat IP dan port untuk setiap perangkat inti Greengrass yang terkait.
- Sertifikat CA perangkat inti, yang dapat digunakan oleh perangkat klien untuk mengautentikasi perangkat inti Greengrass.

> **Note**
>
> Perangkat klien juga dapat menggunakan klien penemuan di AWS IoT Device SDK untuk menemukan informasi konektivitas untuk perangkat inti Greengrass. Klien penemuan menggunakan API penemuan. Untuk informasi selengkapnya, lihat hal berikut:
>
> - Uji komunikasi perangkat klien
> - Greengrass Discovery RESTful API di Panduan Developer AWS IoT Greengrass Version 1.

Untuk menggunakan operasi API ini, kirim permintaan HTTP untuk API penemuan tersebut pada titik akhir bidang data Greengrass. Titik akhir API ini memiliki format berikut.

```
https://greengrass-ats.iot.region.amazonaws.com:port/greengrass/discover/thing/thing-name
```

Untuk daftar yang didukung Wilayah AWS dan titik akhir untuk API AWS IoT Greengrass penemuan, lihat AWS IoT Greengrass V2titik akhir dan kuota di. Referensi Umum AWS Operasi API ini tersedia

hanya pada titik akhir bidang data Greengrass. Titik akhir bidang kontrol yang Anda gunakan untuk mengelola komponen dan deployment berbeda dari titik akhir bidang data.

> **Note**
>
> API penemuan adalah sama untuk AWS IoT Greengrass V1 dan AWS IoT Greengrass V2. Jika Anda memiliki perangkat klien yang terhubung ke AWS IoT Greengrass V1 inti, Anda dapat menghubungkannya ke perangkat AWS IoT Greengrass V2 inti tanpa mengubah kode pada perangkat klien. Untuk informasi selengkapnya, lihat Greengrass Discovery RESTful API di Panduan Developer AWS IoT Greengrass Version 1.

Topik

Autentikasi dan otorisasi penemuan

Untuk menggunakan API penemuan untuk mengambil informasi konektivitas, perangkat klien harus menggunakan autentikasi bersama TLS dengan sertifikat klien X.509 untuk mengautentikasi. Untuk informasi lebih lanjut, lihat Sertifikat klien X.509 di Panduan Developer AWS IoT Core.

Perangkat klien juga harus memiliki izin untuk melakukan tindakan `greengrass:Discover`. Contoh kebijakan AWS IoT berikut memungkinkan objek AWS IoT bernama `MyClientDevice1`untuk melakukan `Discover` untuk dirinya sendiri.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "greengrass:Discover",
      "Resource": [
        "arn:aws:iot:us-west-2:123456789012:thing/MyClientDevice1"
      ]
    }
```

```
  ]
}
```

> **Important**
>
> Variabel kebijakan objek (`iot:Connection.Thing.*`) tidak didukung di kebijakan AWS IoT untuk perangkat inti atau operasi bidang data Greengrass. Sebaliknya, Anda dapat menggunakan wildcard yang cocok dengan beberapa perangkat yang memiliki nama yang sama. Misalnya, Anda dapat menentukan `MyGreengrassDevice*` agar cocok dengan `MyGreengrassDevice1`, `MyGreengrassDevice2`, dan sebagainya.

Untuk informasi selengkapnya, lihat kebijakan AWS IoT Core di Panduan Developer AWS IoT Core.

Permintaan

Permintaan berisi header HTTP standar dan dikirim ke titik akhir penemuan Greengrass, seperti yang ditunjukkan dalam contoh berikut.

Nomor port tergantung pada apakah perangkat inti dikonfigurasi untuk mengirim lalu lintas HTTPS melalui port 8443 atau port 443. Untuk informasi selengkapnya, lihat the section called “Hubungkan pada port 443 atau melalui proksi jaringan”.

> **Note**
>
> Contoh ini menggunakan endpoint Amazon Trust Services (ATS), yang bekerja dengan sertifikat CA akar ATS yang disarankan. Titik akhir harus sesuai dengan jenis sertifikat CA akar.

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
```

> **Note**
>
> Klien yang terhubung pada port 443 harus menerapkan ekstensi TLS Application Layer Protocol Negotiation (ALPN) dan melewati `x-amzn-http-ca` sebagai `ProtocolName` di `ProtocolNameList`. Untuk informasi selengkapnya, lihat Protokol dalam Panduan Developer AWS IoT.

Respons

Setelah berhasil, header respons menyertakan kode status HTTP 200 dan badan respons berisi dokumen respons temukan.

> **Note**
>
> Karena AWS IoT Greengrass V2 menggunakan API penemuan yang sama seperti AWS IoT Greengrass V1, responnya akan mengatur informasi sesuai dengan konsep AWS IoT Greengrass V1, seperti grup Greengrass. Tanggapan berisi daftar grup Greengrass. Di AWS IoT Greengrass V2, setiap perangkat inti berada dalam grupnya sendiri, di mana grup hanya berisi perangkat inti dan informasi konektivitas.

Contoh dokumen respon temuan

Dokumen berikut menunjukkan respon untuk perangkat klien yang terkait dengan satu perangkat inti Greengrass. Perangkat inti memiliki satu titik akhir dan satu sertifikat CA.

```
{
  "GGGroups": [
    {
      "GGGroupId": "greengrassV2-coreDevice-core-device-01-thing-name",
      "Cores": [
        {
          "thingArn": "core-device-01-thing-arn",
          "Connectivity": [
            {
              "id": "core-device-01-connection-id",
              "hostAddress": "core-device-01-address",
              "portNumber": core-device-01-port,
              "metadata": "core-device-01-description"
```

```
            }
          ]
        }
      ],
      "CAs": [
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----"
      ]
    }
  ]
}
```

Dokumen berikut menunjukkan respons untuk perangkat klien yang terkait dengan dua perangkat inti. Perangkat inti memiliki beberapa titik akhir dan beberapa sertifikat CA grup.

```
{
  "GGGroups": [
    {
      "GGGroupId": "greengrassV2-coreDevice-core-device-01-thing-name",
      "Cores": [
        {
          "thingArn": "core-device-01-thing-arn",
          "Connectivity": [
            {
              "id": "core-device-01-connection-id",
              "hostAddress": "core-device-01-address",
              "portNumber": core-device-01-port,
              "metadata": "core-device-01-connection-1-description"
            },
            {
              "id": "core-device-01-connection-id-2",
              "hostAddress": "core-device-01-address-2",
              "portNumber": core-device-01-port-2,
              "metadata": "core-device-01-connection-2-description"
            }
          ]
        }
      ],
      "CAs": [
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----"
      ]
    },
```

```
    {
      "GGGroupId": "greengrassV2-coreDevice-core-device-02-thing-name",
      "Cores": [
        {
          "thingArn":"core-device-02-thing-arn",
          "Connectivity" : [
            {
              "id": "core-device-02-connection-id",
              "hostAddress": "core-device-02-address",
              "portNumber": core-device-02-port,
              "metadata": "core-device-02-connection-1-description"
            }
          ]
        }
      ],
      "CAs": [
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----"
      ]
    }
  ]
}
```

Uji API penemuan dengan cURL

Jika Anda memiliki cURL yang diinstal, Anda dapat menguji API penemuan. Contoh berikut menentukan sertifikat perangkat klien untuk mengautentikasi permintaan ke titik akhir penemuan API Greengrass.

```
curl -i \
  --cert 1a23bc4d56.cert.pem \
  --key 1a23bc4d56.private.key \
  https://greengrass-ats.iot.us-west-2.amazonaws.com:8443/greengrass/discover/
thing/MyClientDevice1
```

> **Note**
>
> Argumen `-i` menentukan header tanggapan HTTP output. Anda dapat menggunakan opsi ini untuk membantu mengidentifikasi kesalahan.

Jika permintaan berhasil, perintah ini menghasilkan tanggapan yang serupa dengan contoh berikut.

```
{
  "GGGroups": [
    {
      "GGGroupId": "greengrassV2-coreDevice-MyGreengrassCore",
      "Cores": [
        {
          "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
          "Connectivity": [
            {
              "Id": "AUTOIP_192.168.1.4_1",
              "HostAddress": "192.168.1.5",
              "PortNumber": 8883,
              "Metadata": ""
            }
          ]
        }
      ],
      "CAs": [
        "-----BEGIN CERTIFICATE-----\ncert-contents\n-----END CERTIFICATE-----\n"
      ]
    }
  ]
}
```

Jika aplikasi mengeluarkan kesalahan, lihat Pemecahan masalah penemuan Greengrass.

Relai pesan MQTT antara perangkat klien dan AWS IoT Core

Anda dapat merelai pesan MQTT dan data lainnya antara perangkat klien dan AWS IoT Core. Perangkat klien tersambung ke komponen broker MQTT yang berjalan pada perangkat inti. Secara default, perangkat inti tidak menyampaikan pesan MQTT atau data antara perangkat klien dan AWS IoT Core. Perangkat klien dapat berkomunikasi hanya dengan satu sama lain melalui MQTT secara default.

Untuk menyampaikan pesan MQTT antara perangkat klien dan AWS IoT Core, konfigurasi Komponen jembatan MQTT untuk melakukan hal berikut:

- Relai pesan dari perangkat klien ke AWS IoT Core.
- Relai pesan dari AWS IoT Core ke perangkat klien.

> **Note**
>
> Jembatan MQTT menggunakan QoS 1 untuk mempublikasikan dan berlangganan AWS IoT Core, bahkan ketika perangkat klien menggunakan QoS 0 untuk mempublikasikan dan berlangganan broker MQTT lokal. Sebagai hasilnya, Anda mungkin melihat latensi tambahan ketika Anda menyampaikan pesan MQTT dari perangkat klien pada broker MQTT lokal ke AWS IoT Core. Untuk informasi lebih lanjut tentang konfigurasi MQTT pada perangkat inti, lihat Konfigurasikan pengaturan batas waktu dan cache MQTT.

Topik

Konfigurasikan dan deploy komponen jembatan MQTT

Komponen jembatan MQTT mengonsumsi daftar pemetaan topik yang masing-masing menentukan sumber pesan dan tujuan pesan. Untuk merelai pesan antara perangkat klien dan AWS IoT Core, deploy komponen jembatan MQTT, dan tentukan setiap sumber dan tujuan topik dalam konfigurasi komponen.

Untuk men-deploy komponen jembatan MQTT ke perangkat inti atau kelompok perangkat inti, buat deployment yang mencakup komponen `aws.greengrass.clientdevices.mqtt.Bridge`. Tentukan pemetaan topik,`mqttTopicMapping`, dalam konfigurasi komponen jembatan MQTT dalam penerapan.

Contoh berikut menentukan deployment yang mengonfigurasi komponen jembatan MQTT untuk merelai pesan pada topik yang cocok dengan filter topik `clients/+/hello/world` dari perangkat klien ke AWS IoT Core. Pembaruan konfigurasi `merge` memerlukan objek JSON berserial. Untuk informasi selengkapnya, lihat Perbarui konfigurasi komponen.

Console

```
{
  "mqttTopicMapping": {
    "HelloWorldIotCore": {
      "topic": "clients/+/hello/world",
```

```
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

AWS CLI

```
{
  "components": {
    "aws.greengrass.clientdevices.mqtt.Bridge": {
      "version": "2.0.0",
      "configurationUpdate": {
        "merge": "{\"mqttTopicMapping\":{\"HelloWorldIotCore\":{\"topic\":\"clients/
+/hello/world\",\"source\":\"LocalMqtt\",\"target\":\"IotCore\"}}}"
      }
    }
    ...
  }
}
```

Relai pesan MQTT

Untuk menyampaikan pesan MQTT antara perangkat klien dan AWS IoT Core, konfigurasikan dan deploy komponen jembatan MQTT dan tentukan topik yang akan direlai.

Example Contoh: Relai pesan pada topik dari perangkat klien ke AWS IoT Core

Konfigurasi komponen jembatan MQTT berikut menentukan penyampaian pesan pada topik yang cocok dengan filter topik `clients/+/hello/world/event` dari perangkat klien ke AWS IoT Core.

```
{
  "mqttTopicMapping": {
    "HelloWorldEvent": {
      "topic": "clients/+/hello/world/event",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

Example Contoh: Relai pesan pada topik dari AWS IoT Core ke perangkat klien

Konfigurasi komponen jembatan MQTT berikut menentukan penyampaian pesan pada topik yang cocok dengan filter topik `clients/+/hello/world/event/response` dari AWS IoT Core ke perangkat klien.

```
{
  "mqttTopicMapping": {
    "HelloWorldEventConfirmation": {
      "topic": "clients/+/hello/world/event/response",
      "source": "IotCore",
      "target": "LocalMqtt"
    }
  }
}
```

Berinteraksilah dengan perangkat klien dalam komponen

Anda dapat mengembangkan komponen Greengrass kustom yang berinteraksi dengan perangkat klien yang terhubung ke perangkat inti. Misalnya, Anda dapat mengembangkan komponen yang melakukan hal berikut:

- Bertindaklah atas pesan MQTT dari perangkat klien dan kirim data ke tujuan AWS Cloud.
- Kirim pesan MQTT ke perangkat klien untuk melakukan tindakan.

Perangkat klien terhubung ke dan berkomunikasi dengan perangkat inti melalui komponen broker MQTT yang berjalan pada perangkat inti. Secara default, perangkat klien hanya dapat berkomunikasi satu sama lain melalui MQTT, dan komponen Greengrass tidak dapat menerima pesan MQTT ini atau mengirim pesan ke perangkat klien.

Komponen Greengrass menggunakan antarmuka publish/subscribe lokal untuk berkomunikasi di perangkat inti. Untuk berkomunikasi dengan perangkat klien dalam komponen Greengrass, konfigurasikan komponen jembatan MQTT untuk melakukan hal berikut:

- Relai pesan MQTT dari perangkat klien ke publish/subscribe lokal.
- Relai pesan MQTT dari publish/subscribe lokal ke perangkat klien.

Anda juga dapat berinteraksi dengan bayangan perangkat klien dalam komponen Greengrass. Untuk informasi selengkapnya, lihat Berinteraksi dengan dan menyinkronkan bayangan perangkat klien.

Topik

- Konfigurasikan dan deploy komponen jembatan MQTT
- Terima pesan MQTT dari perangkat klien
- Kirim pesan MQTT ke perangkat klien

Konfigurasikan dan deploy komponen jembatan MQTT

Komponen jembatan MQTT mengonsumsi daftar pemetaan topik yang masing-masing menentukan sumber pesan dan tujuan pesan. Untuk berkomunikasi dengan perangkat klien, deploy komponen jembatan MQTT, dan tentukan setiap sumber dan tujuan topik dalam konfigurasi komponen.

Untuk men-deploy komponen jembatan MQTT ke perangkat inti atau kelompok perangkat inti, buat deployment yang mencakup komponen `aws.greengrass.clientdevices.mqtt.Bridge`. Tentukan pemetaan topik,`mqttTopicMapping`, dalam konfigurasi komponen jembatan MQTT dalam penerapan.

Contoh berikut mendefinisikan deployment yang mengonfigurasi komponen jembatan MQTT untuk merelai topik `clients/MyClientDevice1/hello/world` dari perangkat klien ke broker publish/subscribe lokal. Pembaruan konfigurasi `merge` memerlukan objek JSON berserial. Untuk informasi selengkapnya, lihat Perbarui konfigurasi komponen.

Console

```
{
  "mqttTopicMapping": {
    "HelloWorldPubsub": {
      "topic": "clients/MyClientDevice1/hello/world",
      "source": "LocalMqtt",
      "target": "Pubsub"
    }
  }
}
```

AWS CLI

```
{
```

```
  "components": {
    "aws.greengrass.clientdevices.mqtt.Bridge": {
      "version": "2.0.0",
      "configurationUpdate": {
        "merge": "\"mqttTopicMapping\":{\"HelloWorldPubsub\":{\"topic\":\"clients/
MyClientDevice1/hello/world\",\"source\":\"LocalMqtt\",\"target\":\"Pubsub\"}}}"
      }
    }
    ...
  }
}
```

Anda dapat menggunakan wildcard topik MQTT untuk menyampaikan pesan pada topik yang cocok dengan filter topik. Jika Anda menggunakan MQTT bridge v2.2.0 atau yang lebih baru, Anda dapat menggunakan wildcard topik MQTT di filter topik saat broker sumber menerbitkan/berlangganan lokal. Untuk informasi lebih lanjut, lihat konfigurasi komponen jembatan MQTT.

Terima pesan MQTT dari perangkat klien

Anda dapat berlangganan topik publish/subscribe lokal yang dikonfigurasi untuk komponen jembatan MQTT untuk menerima pesan dari perangkat klien.

Untuk menerima pesan MQTT dari perangkat klien dalam komponen kustom

1. Konfigurasikan dan deploy komponen jembatan MQTT untuk menyampaikan pesan dari topik MQTT di mana perangkat klien mempublikasikan ke topik publish/subscribe lokal.
2. Gunakan antarmuka IPC publish/subscribe lokal untuk berlangganan topik di mana jembatan MQTT merelai pesan. Lihat informasi yang lebih lengkap di Pesan lokal publikasi/berlangganan dan SubscribeToTopic.

Tutorial hubungkan dan uji perangkat klien mencakup bagian di mana Anda mengembangkan komponen yang berlangganan pesan dari perangkat klien. Untuk informasi selengkapnya, lihat Langkah 4: Kembangkan komponen yang berkomunikasi dengan perangkat klien.

Kirim pesan MQTT ke perangkat klien

Anda dapat mempublikasikan topik publish/subscribe lokal yang dikonfigurasi untuk komponen jembatan MQTT untuk menerima pesan dari perangkat klien.

Untuk mempublikasikan pesan MQTT ke perangkat klien dalam komponen kustom

1. Konfigurasikan dan deploy komponen jembatan MQTT untuk menyampaikan pesan dari topik publish/subscribe lokal ke topik MQTT tempat perangkat klien berlangganan.
2. Gunakan antarmuka IPC publish/subscribe lokal untuk mempublikasikan topik di mana jembatan MQTT merelai pesan. Lihat informasi yang lebih lengkap di Pesan lokal publikasi/berlangganan dan PublishToTopic.

Berinteraksi dengan dan menyinkronkan bayangan perangkat klien

Anda dapat menggunakan komponen pengelola bayangan untuk mengelola bayangan lokal, termasuk bayangan perangkat klien. Anda dapat menggunakan shadow manager untuk melakukan hal berikut:

- Berinteraksi dengan bayangan perangkat klien dalam komponen Greengrass.
- Menyinkronkan bayangan perangkat klien dengan AWS IoT Core.

> **Note**
>
> Komponen manajer bayangan tidak menyinkronkan bayangan dengan AWS IoT Core secara default. Anda harus mengonfigurasi komponen manajer bayangan untuk menentukan bayangan perangkat klien yang akan disinkronkan.

Topik

Prasyarat

Untuk berinteraksi dengan bayangan perangkat klien dan menyinkronkan bayangan perangkat klienAWS IoT Core, perangkat inti harus memenuhi persyaratan berikut:

- Perangkat inti harus menjalankan komponen berikut, selain komponen Greengrass untuk dukungan perangkat klien:
  - Greengrass nucleus v2.6.0 atau yang lebih baru
  - Shadow manager v2.2.0 atau yang lebih baru
  - Jembatan MQTT v2.2.0 atau yang lebih baru
- Komponen autentikasi perangkat klien harus dikonfigurasi untuk memungkinkan perangkat klien berkomunikasi pada topik bayangan perangkat.

Aktifkan pengelola bayangan untuk berkomunikasi dengan perangkat klien

Secara default, komponen manajer bayangan tidak mengelola bayangan perangkat klien. Untuk mengaktifkan fitur ini, Anda harus menyampaikan pesan MQTT antara perangkat klien dan komponen shadow manager. Perangkat klien menggunakan pesan MQTT untuk menerima dan mengirim pembaruan bayangan perangkat. Komponen pengelola bayangan berlangganan antarmuka penerbitan/langganan Greengrass lokal, sehingga Anda dapat mengonfigurasi komponen jembatan MQTT untuk menyampaikan pesan MQTT pada topik bayangan perangkat.

Komponen jembatan MQTT mengonsumsi daftar pemetaan topik yang masing-masing menentukan sumber pesan dan tujuan pesan. Untuk mengaktifkan komponen pengelola bayangan mengelola bayangan perangkat klien, gunakan komponen jembatan MQTT, dan tentukan topik bayangan untuk bayangan perangkat klien. Anda harus mengonfigurasi jembatan untuk merelai pesan di kedua arah antara MQTT lokal dan publish/subscribe lokal.

Untuk men-deploy komponen jembatan MQTT ke perangkat inti atau kelompok perangkat inti, buat deployment yang mencakup komponen `aws.greengrass.clientdevices.mqtt.Bridge`. Tentukan pemetaan topik,`mqttTopicMapping`, dalam konfigurasi komponen jembatan MQTT dalam penerapan.

Gunakan contoh berikut untuk mengonfigurasi komponen jembatan MQTT untuk mengaktifkan komunikasi antara perangkat klien dan komponen shadow manager.

> **Note**
>
> Anda dapat menggunakan contoh-contoh konfigurasi ini di konsol AWS IoT Greengrass. Jika Anda menggunakan API AWS IoT Greengrass, pembaruan konfigurasi `mergeakan` memerlukan objek JSON berserial, sehingga Anda harus membuat serial objek JSON berikut ke dalam string. Untuk informasi selengkapnya, lihat Perbarui konfigurasi komponen.

Example Contoh: Kelola semua bayangan perangkat klien

Contoh konfigurasi jembatan MQTT berikut memungkinkan shadow manager untuk mengelola semua bayangan untuk semua perangkat klien.

```
{
  "mqttTopicMapping": {
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Example Contoh: Mengelola bayangan untuk perangkat klien

Contoh konfigurasi jembatan MQTT berikut memungkinkan shadow manager untuk mengelola semua bayangan untuk perangkat klien bernama. `MyClientDevice`

```
{
  "mqttTopicMapping": {
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/MyClientDevice/shadow/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/MyClientDevice/shadow/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Example Contoh: Mengelola bayangan bernama untuk semua perangkat klien

Contoh konfigurasi jembatan MQTT berikut memungkinkan shadow manager untuk mengelola bayangan bernama `DeviceConfiguration` untuk semua perangkat klien.

```
{
  "mqttTopicMapping": {
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/name/DeviceConfiguration/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/name/DeviceConfiguration/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Example Contoh: Kelola bayangan tanpa nama semua perangkat klien

Contoh konfigurasi jembatan MQTT berikut memungkinkan pengelola bayangan untuk mengelola bayangan yang tidak disebutkan namanya, tetapi bukan bayangan bernama, untuk semua perangkat klien.

```
{
  "mqttTopicMapping": {
    "DeleteShadowLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/delete",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "DeleteShadowPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/delete/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    },
    "GetShadowLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/get",
      "source": "LocalMqtt",
      "target": "Pubsub"
```

```
    },
    "GetShadowPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/get/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    },
    "UpdateShadowLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/update",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "UpdateShadowPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/update/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Berinteraksi dengan bayangan perangkat klien dalam komponen

Anda dapat mengembangkan komponen kustom yang menggunakan layanan bayangan lokal untuk membaca dan memodifikasi dokumen bayangan lokal perangkat klien. Untuk informasi selengkapnya, lihat Berinteraksilah dengan bayangan dalam komponen.

Sinkronkan bayangan perangkat klien dengan AWS IoT Core

Anda dapat mengonfigurasi komponen shadow manager untuk menyinkronkan status bayangan perangkat klien lokal denganAWS IoT Core. Lihat informasi yang lebih lengkap di Sinkronkan bayangan perangkat lokal dengan AWS IoT Core.

Menyelesaikan masalah perangkat klien

Gunakan informasi pemecahan masalah dan solusi di bagian ini untuk membantu menyelesaikan masalah dengan perangkat klien Greengrass dan komponen perangkat klien.

Topik

- Masalah penemuan Greengrass
- Masalah koneksi MQTT

Masalah penemuan Greengrass

Gunakan informasi berikut untuk memecahkan masalah dengan penemuan Greengrass. Masalah ini dapat terjadi ketika perangkat klien menggunakan API penemuan Greengrass untuk mengidentifikasi perangkat inti Greengrass yang dapat mereka hubungkan.

Topik

Masalah penemuan Greengrass (HTTP API)

Gunakan informasi berikut untuk memecahkan masalah dengan penemuan Greengrass. Anda mungkin melihat kesalahan ini jika Anda menguji API penemuan dengan cURL.

Topik

curl: (52) Empty reply from server

Anda mungkin melihat kesalahan ini jika Anda menentukan sertifikat AWS IoT dalam permintaan.

Periksa apakah perangkat klien memiliki sertifikat terlampir, dan apakah sertifikat itu aktif. Untuk informasi selengkapnya, lihat Lampirkan objek atau kebijakan ke sertifikat klien dan Aktifkan atau nonaktifkan sertifikat klien dalam Panduan Developer AWS IoT Core.

HTTP 403: {"message":null,"traceId":"a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"}

Anda mungkin melihat kesalahan ini jika perangkat klien tidak memiliki izin untuk memanggil `greengrass:Discover` untuk dirinya sendiri.

Periksa apakah sertifikat perangkat klien memiliki kebijakan yang mengizinkan `greengrass:Discover`. Anda tidak dapat menggunakan variabel kebijakan (`iot:Connection.Thing.*`) di bagian `Resource` untuk izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi penemuan.

HTTP 404: {"errorMessage":"The thing provided for discovery was not found"}

Anda mungkin melihat kesalahan ini dalam kasus berikut:

- Perangkat klien tidak terkait dengan perangkat atau grup inti Greengrass apa pun. AWS IoT Greengrass V1
- Tak satu pun dari perangkat AWS IoT Greengrass V1 atau grup inti Greengrass terkait perangkat klien memiliki titik akhir broker MQTT.
- Tak satu pun dari perangkat inti Greengrass terkait perangkat klien menjalankan komponen autentikasi perangkat klien.

Periksa apakah perangkat klien terkait ke perangkat inti yang Anda inginkan untuk terhubung. Kemudian, periksa apakah perangkat inti menjalankan komponen autentikasi perangkat klien dan memiliki setidaknya satu titik akhir broker MQTT. Untuk informasi selengkapnya, lihat hal berikut:

- Kaitkan perangkat klien
- Kelola titik akhir perangkat inti
- Konfigurasikan penemuan cloud (konsol)

Masalah penemuan Greengrass (AWS IoT Device SDK v2 for Python)

Gunakan informasi berikut untuk memecahkan masalah dengan penemuan Greengrass di AWS IoT Device SDK v2 for Python.

Topik

- awscrt.exceptions.AwsCrtError: AWS_ERROR_HTTP_CONNECTION_CLOSED: The connection has closed or is closing.
- awsiot.greengrass_discovery.DiscoveryException: ('Error during discover call: response_code=403', 403)
- awsiot.greengrass_discovery.DiscoveryException: ('Error during discover call: response_code=404', 404)

awscrt.exceptions.AwsCrtError: AWS_ERROR_HTTP_CONNECTION_CLOSED: The connection has closed or is closing.

Anda mungkin melihat kesalahan ini jika Anda menentukan sertifikat AWS IoT yang tidak aktif dalam permintaan.

Periksa apakah perangkat klien memiliki sertifikat terlampir, dan apakah sertifikat itu aktif. Untuk informasi selengkapnya, lihat Lampirkan objek atau kebijakan ke sertifikat klien dan Aktifkan atau nonaktifkan sertifikat klien dalam Panduan Developer AWS IoT Core.

awsiot.greengrass_discovery.DiscoveryException: ('Error during discover call: response_code=403', 403)

Anda mungkin melihat kesalahan ini jika perangkat klien tidak memiliki izin untuk memanggil `greengrass:Discover` untuk dirinya sendiri.

Periksa apakah sertifikat perangkat klien memiliki kebijakan yang mengizinkan `greengrass:Discover`. Anda tidak dapat menggunakan variabel kebijakan (`iot:Connection.Thing.*`) di bagian `Resource` untuk izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi penemuan.

awsiot.greengrass_discovery.DiscoveryException: ('Error during discover call: response_code=404', 404)

Anda mungkin melihat kesalahan ini dalam kasus berikut:

- Perangkat klien tidak terkait dengan perangkat atau grup inti Greengrass apa pun. AWS IoT Greengrass V1
- Tak satu pun dari perangkat AWS IoT Greengrass V1 atau grup inti Greengrass terkait perangkat klien memiliki titik akhir broker MQTT.
- Tak satu pun dari perangkat inti Greengrass terkait perangkat klien menjalankan komponen autentikasi perangkat klien.

Periksa apakah perangkat klien terkait ke perangkat inti yang Anda inginkan untuk terhubung. Kemudian, periksa apakah perangkat inti menjalankan komponen autentikasi perangkat klien dan memiliki setidaknya satu titik akhir broker MQTT. Untuk informasi selengkapnya, lihat hal berikut:

- Kaitkan perangkat klien
- Kelola titik akhir perangkat inti

- Konfigurasikan penemuan cloud (konsol)

Masalah penemuan Greengrass (AWS IoT Device SDK v2 for C++)

Gunakan informasi berikut untuk memecahkan masalah dengan penemuan Greengrass di AWS IoT Device SDK v2 for C++.

Topik

- aws-c-http: AWS_ERROR_HTTP_CONNECTION_CLOSED, The connection has closed or is closing.
- aws-c-common: AWS_ERROR_UNKNOWN, Unknown error. (HTTP 403)
- aws-c-common: AWS_ERROR_UNKNOWN, Unknown error. (HTTP 404)

aws-c-http: AWS_ERROR_HTTP_CONNECTION_CLOSED, The connection has closed or is closing.

Anda mungkin melihat kesalahan ini jika Anda menentukan sertifikat AWS IoT yang tidak aktif dalam permintaan.

Periksa apakah perangkat klien memiliki sertifikat terlampir, dan apakah sertifikat itu aktif. Untuk informasi selengkapnya, lihat Lampirkan objek atau kebijakan ke sertifikat klien dan Aktifkan atau nonaktifkan sertifikat klien dalam Panduan Developer AWS IoT Core.

aws-c-common: AWS_ERROR_UNKNOWN, Unknown error. (HTTP 403)

Anda mungkin melihat kesalahan ini jika perangkat klien tidak memiliki izin untuk memanggil `greengrass:Discover` untuk dirinya sendiri.

Periksa apakah sertifikat perangkat klien memiliki kebijakan yang mengizinkan `greengrass:Discover`. Anda tidak dapat menggunakan variabel kebijakan (`iot:Connection.Thing.*`) di bagian `Resource` untuk izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi penemuan.

aws-c-common: AWS_ERROR_UNKNOWN, Unknown error. (HTTP 404)

Anda mungkin melihat kesalahan ini dalam kasus berikut:

- Perangkat klien tidak terkait dengan perangkat atau grup inti Greengrass apa pun. AWS IoT Greengrass V1

- Tak satu pun dari perangkat AWS IoT Greengrass V1 atau grup inti Greengrass terkait perangkat klien memiliki titik akhir broker MQTT.
- Tak satu pun dari perangkat inti Greengrass terkait perangkat klien menjalankan komponen autentikasi perangkat klien.

Periksa apakah perangkat klien terkait ke perangkat inti yang Anda inginkan untuk terhubung. Kemudian, periksa apakah perangkat inti menjalankan komponen autentikasi perangkat klien dan memiliki setidaknya satu titik akhir broker MQTT. Untuk informasi selengkapnya, lihat hal berikut:

- Kaitkan perangkat klien
- Kelola titik akhir perangkat inti
- Konfigurasikan penemuan cloud (konsol)

Masalah penemuan Greengrass (v2 untuk) AWS IoT Device SDK JavaScript

Gunakan informasi berikut untuk memecahkan masalah dengan penemuan Greengrass di v2 untuk. AWS IoT Device SDK JavaScript

Topik

- Error: aws-c-http: AWS_ERROR_HTTP_CONNECTION_CLOSED, The connection has closed or is closing.
- Error: Discovery failed (headers: [object Object]) { response_code: 403 }
- Error: Discovery failed (headers: [object Object]) { response_code: 404 }
- Error: Discovery failed (headers: [object Object])

Error: aws-c-http: AWS_ERROR_HTTP_CONNECTION_CLOSED, The connection has closed or is closing.

Anda mungkin melihat kesalahan ini jika Anda menentukan sertifikat AWS IoT yang tidak aktif dalam permintaan.

Periksa apakah perangkat klien memiliki sertifikat terlampir, dan apakah sertifikat itu aktif. Untuk informasi selengkapnya, lihat Lampirkan objek atau kebijakan ke sertifikat klien dan Aktifkan atau nonaktifkan sertifikat klien dalam Panduan Developer AWS IoT Core.

Error: Discovery failed (headers: [object Object]) { response_code: 403 }

Anda mungkin melihat kesalahan ini jika perangkat klien tidak memiliki izin untuk memanggil `greengrass:Discover` untuk dirinya sendiri.

Periksa apakah sertifikat perangkat klien memiliki kebijakan yang mengizinkan `greengrass:Discover`. Anda tidak dapat menggunakan variabel kebijakan (`iot:Connection.Thing.*`) di bagian `Resource` untuk izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi penemuan.

Error: Discovery failed (headers: [object Object]) { response_code: 404 }

Anda mungkin melihat kesalahan ini dalam kasus berikut:

- Perangkat klien tidak terkait dengan perangkat atau grup inti Greengrass apa pun. AWS IoT Greengrass V1
- Tak satu pun dari perangkat AWS IoT Greengrass V1 atau grup inti Greengrass terkait perangkat klien memiliki titik akhir broker MQTT.
- Tak satu pun dari perangkat inti Greengrass terkait perangkat klien menjalankan komponen autentikasi perangkat klien.

Periksa apakah perangkat klien terkait ke perangkat inti yang Anda inginkan untuk terhubung. Kemudian, periksa apakah perangkat inti menjalankan komponen autentikasi perangkat klien dan memiliki setidaknya satu titik akhir broker MQTT. Untuk informasi selengkapnya, lihat hal berikut:

- Kaitkan perangkat klien
- Kelola titik akhir perangkat inti
- Konfigurasikan penemuan cloud (konsol)

Error: Discovery failed (headers: [object Object])

Anda mungkin melihat kesalahan ini (tanpa kode respons HTTP) ketika Anda menjalankan sampel penemuan Greengrass. Kesalahan ini dapat terjadi karena berbagai alasan.

- Anda mungkin melihat kesalahan ini jika perangkat klien tidak memiliki izin untuk memanggil `greengrass:Discover` untuk dirinya sendiri.

  Periksa apakah sertifikat perangkat klien memiliki kebijakan yang mengizinkan `greengrass:Discover`. Anda tidak dapat menggunakan variabel kebijakan

(`iot:Connection.Thing.*`) di bagian `Resource` untuk izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi penemuan.

- Anda mungkin melihat kesalahan ini dalam kasus berikut:
  - Perangkat klien tidak terkait dengan perangkat atau grup inti Greengrass apa pun. AWS IoT Greengrass V1
  - Tak satu pun dari perangkat AWS IoT Greengrass V1 atau grup inti Greengrass terkait perangkat klien memiliki titik akhir broker MQTT.
  - Tak satu pun dari perangkat inti Greengrass terkait perangkat klien menjalankan komponen autentikasi perangkat klien.

  Periksa apakah perangkat klien terkait ke perangkat inti yang Anda inginkan untuk terhubung. Kemudian, periksa apakah perangkat inti menjalankan komponen autentikasi perangkat klien dan memiliki setidaknya satu titik akhir broker MQTT. Untuk informasi selengkapnya, lihat hal berikut:
  - Kaitkan perangkat klien
  - Kelola titik akhir perangkat inti
  - Konfigurasikan penemuan cloud (konsol)

Masalah penemuan Greengrass (v2 for Java AWS IoT Device SDK)

Gunakan informasi berikut untuk memecahkan masalah dengan penemuan Greengrass di AWS IoT Device SDK v2 for Java.

Topik

- software.amazon.awssdk.crt.CrtRuntimeException: Error Getting Response Status Code from HttpStream. (aws_last_error: AWS_ERROR_HTTP_DATA_NOT_AVAILABLE(2062), This data is not yet available.)
- java.lang.RuntimeException: Error x-amzn-ErrorType(403)
- java.lang.RuntimeException: Error x-amzn-ErrorType(404)

software.amazon.awssdk.crt.CrtRuntimeException: Error Getting Response Status Code from HttpStream. (aws_last_error: AWS_ERROR_HTTP_DATA_NOT_AVAILABLE(2062), This data is not yet available.)

Anda mungkin melihat kesalahan ini jika Anda menentukan sertifikat AWS IoT yang tidak aktif dalam permintaan.

Periksa apakah perangkat klien memiliki sertifikat terlampir, dan apakah sertifikat itu aktif. Untuk informasi selengkapnya, lihat Lampirkan objek atau kebijakan ke sertifikat klien dan Aktifkan atau nonaktifkan sertifikat klien dalam Panduan Developer AWS IoT Core.

java.lang.RuntimeException: Error x-amzn-ErrorType(403)

Anda mungkin melihat kesalahan ini jika perangkat klien tidak memiliki izin untuk memanggil `greengrass:Discover` untuk dirinya sendiri.

Periksa apakah sertifikat perangkat klien memiliki kebijakan yang mengizinkan `greengrass:Discover`. Anda tidak dapat menggunakan variabel kebijakan (`iot:Connection.Thing.*`) di bagian `Resource` untuk izin ini. Untuk informasi selengkapnya, lihat Autentikasi dan otorisasi penemuan.

java.lang.RuntimeException: Error x-amzn-ErrorType(404)

Anda mungkin melihat kesalahan ini dalam kasus berikut:

- Perangkat klien tidak terkait dengan perangkat atau grup inti Greengrass apa pun. AWS IoT Greengrass V1
- Tak satu pun dari perangkat AWS IoT Greengrass V1 atau grup inti Greengrass terkait perangkat klien memiliki titik akhir broker MQTT.
- Tak satu pun dari perangkat inti Greengrass terkait perangkat klien menjalankan komponen autentikasi perangkat klien.

Periksa apakah perangkat klien terkait ke perangkat inti yang Anda inginkan untuk terhubung. Kemudian, periksa apakah perangkat inti menjalankan komponen autentikasi perangkat klien dan memiliki setidaknya satu titik akhir broker MQTT. Untuk informasi selengkapnya, lihat hal berikut:

- Kaitkan perangkat klien
- Kelola titik akhir perangkat inti
- Konfigurasikan penemuan cloud (konsol)

Masalah koneksi MQTT

Gunakan informasi berikut untuk memecahkan masalah dengan koneksi MQTT perangkat klien. Masalah ini dapat terjadi ketika perangkat klien mencoba terhubung ke perangkat inti melalui MQTT.

Topik

io.moquette.broker.Authorizator: Client does not have read permissions on the topic

Anda mungkin melihat kesalahan ini di log Greengrass saat perangkat klien mencoba berlangganan topik MQTT yang tidak memiliki izin. Pesan kesalahan mencakup topik.

Periksa apakah konfigurasi komponen autentikasi perangkat klien mencakup yang berikut ini:

- Grup perangkat yang cocok dengan perangkat klien.
- Kebijakan otorisasi perangkat klien untuk grup perangkat yang memberikan `mqtt:subscribe` izin untuk topik tersebut.

Untuk informasi selengkapnya tentang cara menerapkan dan mengonfigurasi komponen autentikasi perangkat klien, lihat berikut ini:

- Konfigurasikan penemuan cloud (konsol)
- Auth perangkat klien
- Buat deployment

Masalah koneksi MQTT (Python)

Gunakan informasi berikut untuk memecahkan masalah dengan koneksi MQTT perangkat klien saat Anda menggunakan v2 untuk Python. AWS IoT Device SDK

Topik

AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred

Anda mungkin melihat kesalahan ini jika komponen autentikasi perangkat klien tidak menentukan kebijakan otorisasi perangkat klien yang memberikan izin perangkat klien untuk terhubung.

Periksa apakah konfigurasi komponen autentikasi perangkat klien menyertakan yang berikut ini:

- Grup perangkat yang cocok dengan perangkat klien.
- Kebijakan otorisasi perangkat klien untuk grup perangkat yang memberikan `mqtt:connect` izin untuk perangkat klien.

Untuk informasi selengkapnya tentang cara menerapkan dan mengonfigurasi komponen autentikasi perangkat klien, lihat berikut ini:

- Konfigurasikan penemuan cloud (konsol)
- Auth perangkat klien
- Buat deployment

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Anda mungkin melihat kesalahan ini jika komponen autentikasi perangkat klien tidak menentukan kebijakan otorisasi perangkat klien yang memberikan izin perangkat klien untuk terhubung.

Periksa apakah konfigurasi komponen autentikasi perangkat klien menyertakan yang berikut ini:

- Grup perangkat yang cocok dengan perangkat klien.
- Kebijakan otorisasi perangkat klien untuk grup perangkat yang memberikan `mqtt:connect` izin untuk perangkat klien.

Untuk informasi selengkapnya tentang cara menerapkan dan mengonfigurasi komponen autentikasi perangkat klien, lihat berikut ini:

- Konfigurasikan penemuan cloud (konsol)
- Auth perangkat klien
- Buat deployment

Masalah koneksi MQTT (C ++)

Gunakan informasi berikut untuk memecahkan masalah dengan koneksi MQTT perangkat klien saat Anda menggunakan v2 untuk C++. AWS IoT Device SDK

Topik

AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred

Anda mungkin melihat kesalahan ini jika komponen autentikasi perangkat klien tidak menentukan kebijakan otorisasi perangkat klien yang memberikan izin perangkat klien untuk terhubung.

Periksa apakah konfigurasi komponen autentikasi perangkat klien menyertakan yang berikut ini:

- Grup perangkat yang cocok dengan perangkat klien.
- Kebijakan otorisasi perangkat klien untuk grup perangkat yang memberikan `mqtt:connect` izin untuk perangkat klien.

Untuk informasi selengkapnya tentang cara menerapkan dan mengonfigurasi komponen autentikasi perangkat klien, lihat berikut ini:

- Konfigurasikan penemuan cloud (konsol)
- Auth perangkat klien
- Buat deployment

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Anda mungkin melihat kesalahan ini jika komponen autentikasi perangkat klien tidak menentukan kebijakan otorisasi perangkat klien yang memberikan izin perangkat klien untuk terhubung.

Periksa apakah konfigurasi komponen autentikasi perangkat klien menyertakan yang berikut ini:

- Grup perangkat yang cocok dengan perangkat klien.
- Kebijakan otorisasi perangkat klien untuk grup perangkat yang memberikan `mqtt:connect` izin untuk perangkat klien.

Untuk informasi selengkapnya tentang cara menerapkan dan mengonfigurasi komponen autentikasi perangkat klien, lihat berikut ini:

- Konfigurasikan penemuan cloud (konsol)
- Auth perangkat klien
- Buat deployment

Masalah koneksi MQTT (Java)

Gunakan informasi berikut untuk memecahkan masalah dengan koneksi MQTT perangkat klien saat Anda menggunakan v2 untuk Java. AWS IoT Device SDK

Topik

- software.amazon.awssdk.crt.mqtt.MqttException: Protocol error occurred
- AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

software.amazon.awssdk.crt.mqtt.MqttException: Protocol error occurred

Anda mungkin melihat kesalahan ini jika komponen autentikasi perangkat klien tidak menentukan kebijakan otorisasi perangkat klien yang memberikan izin perangkat klien untuk terhubung.

Periksa apakah konfigurasi komponen autentikasi perangkat klien menyertakan yang berikut ini:

- Grup perangkat yang cocok dengan perangkat klien.
- Kebijakan otorisasi perangkat klien untuk grup perangkat yang memberikan `mqtt:connect` izin untuk perangkat klien.

Untuk informasi selengkapnya tentang cara menerapkan dan mengonfigurasi komponen autentikasi perangkat klien, lihat berikut ini:

- Konfigurasikan penemuan cloud (konsol)
- Auth perangkat klien
- Buat deployment

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Anda mungkin melihat kesalahan ini jika komponen autentikasi perangkat klien tidak menentukan kebijakan otorisasi perangkat klien yang memberikan izin perangkat klien untuk terhubung.

Periksa apakah konfigurasi komponen autentikasi perangkat klien menyertakan yang berikut ini:

- Grup perangkat yang cocok dengan perangkat klien.
- Kebijakan otorisasi perangkat klien untuk grup perangkat yang memberikan `mqtt:connect` izin untuk perangkat klien.

Untuk informasi selengkapnya tentang cara menerapkan dan mengonfigurasi komponen autentikasi perangkat klien, lihat berikut ini:

- Konfigurasikan penemuan cloud (konsol)
- Auth perangkat klien
- Buat deployment

Masalah koneksi MQTT () JavaScript

Gunakan informasi berikut untuk memecahkan masalah dengan koneksi MQTT perangkat klien saat Anda menggunakan v2 untuk. AWS IoT Device SDK JavaScript

Topik

AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred

Anda mungkin melihat kesalahan ini jika komponen autentikasi perangkat klien tidak menentukan kebijakan otorisasi perangkat klien yang memberikan izin perangkat klien untuk terhubung.

Periksa apakah konfigurasi komponen autentikasi perangkat klien menyertakan yang berikut ini:

- Grup perangkat yang cocok dengan perangkat klien.
- Kebijakan otorisasi perangkat klien untuk grup perangkat yang memberikan `mqtt:connect` izin untuk perangkat klien.

Untuk informasi selengkapnya tentang cara menerapkan dan mengonfigurasi komponen autentikasi perangkat klien, lihat berikut ini:

- Konfigurasikan penemuan cloud (konsol)
- Auth perangkat klien
- Buat deployment

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Anda mungkin melihat kesalahan ini jika komponen autentikasi perangkat klien tidak menentukan kebijakan otorisasi perangkat klien yang memberikan izin perangkat klien untuk terhubung.

Periksa apakah konfigurasi komponen autentikasi perangkat klien menyertakan yang berikut ini:

- Grup perangkat yang cocok dengan perangkat klien.
- Kebijakan otorisasi perangkat klien untuk grup perangkat yang memberikan `mqtt:connect` izin untuk perangkat klien.

Untuk informasi selengkapnya tentang cara menerapkan dan mengonfigurasi komponen autentikasi perangkat klien, lihat berikut ini:

- Konfigurasikan penemuan cloud (konsol)
- Auth perangkat klien
- Buat deployment

Berinteraksilah dengan bayangan perangkat

Perangkat inti Greengrass dapat berinteraksi AWS IoT dengan bayangan perangkat menggunakan komponen. Bayangan adalah dokumen JSON yang menyimpan informasi keadaan saat ini atau yang diinginkan untuk objek AWS IoT. Bayangan dapat membuat keadaan perangkat tersedia untuk komponen AWS IoT Greengrass lainnya entah perangkat tersebut tersambung ke AWS IoTatau tidak. Setiap perangkat AWS IoT memiliki bayangan klasik, yang tidak bernama miliknya sendiri. Anda juga dapat membuat beberapa bayangan bernama untuk setiap perangkat.

Perangkat dan layanan dapat membuat, memperbarui, dan menghapus bayangan cloud dengan menggunakan MQTT dan topik bayangan MQTT yang dipesan, HTTP yang menggunakan Device Shadow REST API, dan AWS CLI untuk AWS IoT.

Komponen shadow manager memungkinkan komponen Greengrass Anda untuk membuat, memperbarui, dan menghapus bayangan lokal dengan menggunakan layanan bayangan lokaldan topik bayangan publikasi/berlangganan lokal. Komponen shadow manager mengelola penyimpanan dokumen bayangan ini pada perangkat inti Anda, dan menangani sinkronisasi informasi keadaan bayangan dengan bayangan cloud.

Anda juga dapat menggunakan komponen shadow manager untuk mengelola bayangan lokal untuk perangkat klien yang terhubung ke perangkat inti. Untuk mengaktifkan shadow manager mengelola bayangan perangkat klien, Anda mengonfigurasi komponen jembatan MQTT untuk menyampaikan pesan antara broker MQTT lokal dan layanan penerbitan/langganan lokal. Untuk informasi selengkapnya, lihat Berinteraksi dengan dan menyinkronkan bayangan perangkat klien.

Untuk informasi lebih lanjut tentang konsep bayangan perangkat AWS IoT, lihat layanan Bayangan Perangkat AWS IoT di Panduan Developer AWS IoT.

Topik

- Berinteraksilah dengan bayangan dalam komponen
- Sinkronkan bayangan perangkat lokal dengan AWS IoT Core

Berinteraksilah dengan bayangan dalam komponen

Anda dapat mengembangkan komponen kustom, termasuk komponen fungsi Lambda, yang menggunakan layanan bayangan lokal untuk membaca dan memodifikasi dokumen bayangan lokal dan dokumen bayangan perangkat klien.

Komponen kustom berinteraksi dengan layanan bayangan lokal dengan menggunakan pustaka inti IPC AWS IoT Greengrass di AWS IoT Device SDK. Komponen shadow manager memungkinkan layanan bayangan lokal pada perangkat inti Anda.

Untuk men-deploy komponen shadow manager pada perangkat inti Greengrass, buat deployment yang meliputi komponen `aws.greengrass.ShadowManager`.

> **Note**
>
> Secara default, deployment komponen shadow manager akan memungkinkan operasi bayangan lokal saja. AWS IoT GreengrassUntuk mengaktifkan sinkronisasi informasi status bayangan untuk bayangan perangkat inti atau bayangan apa pun untuk perangkat klien ke dokumen bayangan awan yang sesuaiAWS IoT Core, Anda harus membuat pembaruan konfigurasi untuk komponen pengelola bayangan yang menyertakan `synchronize` parameter. Untuk informasi selengkapnya, lihat Sinkronkan bayangan perangkat lokal dengan AWS IoT Core.

Topik

Ambil dan memodifikasi keadaan bayangan

Operasi bayangan IPC mengambil dan memperbarui informasi keadaan dalam dokumen bayangan lokal. Komponen shadow manager menangani penyimpanan dokumen bayangan ini pada perangkat inti Anda.

Untuk mengubah keadaan bayangan lokal

1. Tambahkan kebijakan otorisasi ke resep komponen kustom Anda agar komponen dapat menerima pesan tentang topik bayangan lokal.

   Misalnya kebijakan otorisasi, lihat Contoh kebijakan otorisasi IPC bayangan lokal.
2. Gunakan operasi bayangan IPC untuk mengambil dan memodifikasi informasi keadaan bayangan. Untuk informasi selengkapnya tentang penggunaan operasi IPC bayangan dalam kode komponen, lihat Berinteraksi dengan bayangan lokal.

> **Note**
>
> Untuk mengaktifkan perangkat inti berinteraksi dengan bayangan perangkat klien, Anda juga harus mengonfigurasi dan menerapkan komponen jembatan MQTT. Untuk informasi selengkapnya, lihat Mengaktifkan pengelola bayangan untuk berkomunikasi dengan perangkat klien.

Bereaksilah terhadap perubahan keadaan bayangan

Komponen Greengrass menggunakan antarmuka publish/subscribe lokal untuk berkomunikasi di perangkat inti. Untuk mengaktifkan komponen kustom untuk bereaksi terhadap perubahan keadaan bayangan, Anda dapat berlangganan topik publikasi/berlangganan lokal. Hal ini memungkinkan komponen untuk menerima pesan pada topik bayangan lokal, dan kemudian bertindak pada pesan tersebut.

Topik bayangan lokal menggunakan format yang sama seperti topik MQTT bayangan perangkat AWS IoT. Untuk informasi selengkapnya tentang topik bayangan, lihat topik MQTT Bayangan Perangkat di Panduan Developer AWS IoT.

Untuk bereaksi terhadap perubahan keadaan bayangan

1. Tambahkan kebijakan kontrol akses ke resep untuk komponen kustom Anda untuk memungkinkan komponen untuk menerima pesan pada topik bayangan lokal.

   Misalnya kebijakan otorisasi, lihat Contoh kebijakan otorisasi IPC bayangan lokal.
2. Untuk memulai tindakan kustom dalam sebuah komponen, gunakan operasi IPC `SubscribeToTopic` untuk berlangganan topik bayangan tempat Anda ingin menerima pesan. Untuk informasi selengkapnya tentang penggunaan operasi IPC publikasi/berlangganan lokal dalam kode komponen, lihat Pesan lokal publikasi/berlangganan.
3. Untuk memanggil fungsi Lambda, gunakan konfigurasi sumber peristiwa untuk memberikan nama topik bayangan dan menentukan bahwa ia merupakan topik publikasi/berlangganan lokal. Untuk informasi selengkapnya tentang cara membuat komponen fungsi Lambda, lihat Jalankan fungsi AWS Lambda.

> **Note**
>
> Untuk mengaktifkan perangkat inti berinteraksi dengan bayangan perangkat klien, Anda juga harus mengonfigurasi dan menerapkan komponen jembatan MQTT. Untuk informasi selengkapnya, lihat Mengaktifkan pengelola bayangan untuk berkomunikasi dengan perangkat klien.

Sinkronkan bayangan perangkat lokal dengan AWS IoT Core

Komponen shadow manager memungkinkan AWS IoT Greengrass untuk menyinkronkan perangkat lokal bayangan keadaan dengan AWS IoT Core. Anda harus mengubah konfigurasi komponen shadow manager untuk menyertakan parameter konfigurasi `synchronization`, dan menentukan nama objek AWS IoT untuk perangkat Anda, dan bayangan yang ingin Anda sinkronkan.

Ketika Anda mengonfigurasi bayangan manajer untuk menyinkronkan bayangan, ia menyinkronkan semua perubahan keadaan untuk bayangan tertentu, terlepas dari apakah perubahan tersebut terjadi dalam dokumen bayangan lokal atau di dokumen bayangan cloud.

Anda juga dapat menentukan apakah komponen pengelola bayangan menyinkronkan bayangan secara real time atau pada interval periodik. Secara default, komponen pengelola bayangan menyinkronkan bayangan secara real time, sehingga perangkat inti mengirim dan menerima pembaruan bayangan ke dan dari AWS IoT Core saat setiap pembaruan terjadi. Anda dapat mengonfigurasi interval berkala untuk mengurangi penggunaan dan biaya bandwidth.

Topik

Prasyarat

Untuk menyinkronkan bayangan lokal denganAWS IoT Core, Anda harus mengonfigurasi kebijakan perangkat inti Greengrass untuk mengizinkan AWS IoT tindakan kebijakan bayangan berikut. AWS IoT Core

- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

Untuk informasi selengkapnya, lihat hal berikut:

- AWS IoT Coretindakan kebijakan dalam Panduan AWS IoT Pengembang
- Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2
- Memperbarui AWS IoT kebijakan perangkat inti

Konfigurasikan komponen manajer bayangan

Manajer bayangan memerlukan daftar bayangan nama pemetaan untuk menyinkronkan informasi keadaan bayangan dalam dokumen bayangan lokal ke dokumen bayangan cloud di AWS IoT Core.

Untuk menyinkronkan bayangan keadaan, buat deployment yang mencakup komponen `aws.greengrass.ShadowManager`, dan tentukan bayangan yang ingin Anda sinkronkan di parameter konfigurasi `synchronize` dalam konfigurasi shadow manager dalam deployment tersebut.

> **Note**
>
> Untuk mengaktifkan perangkat inti berinteraksi dengan bayangan perangkat klien, Anda juga harus mengonfigurasi dan menerapkan komponen jembatan MQTT. Untuk informasi selengkapnya, lihat Mengaktifkan pengelola bayangan untuk berkomunikasi dengan perangkat klien.

Pembaruan konfigurasi contoh berikut menginstruksikan komponen manajer bayangan untuk menyinkronkan bayangan berikut dengan AWS IoT Core:

- Bayangan klasik untuk perangkat inti
- `MyCoreShadow` yang bernama untuk perangkat inti
- Bayangan klasik untuk objek IoT bernama `MyDevice2`
- Bayangan yang bernama `MyShadowA` dan `MyShadowB` untuk objek IoT bernama `MyDevice1`

Pembaruan konfigurasi ini menentukan untuk menyinkronkan bayangan dengan AWS IoT Core secara real time. Jika Anda menggunakan shadow manager v2.1.0 atau yang lebih baru, Anda dapat mengonfigurasi komponen shadow manager untuk menyinkronkan bayangan pada interval periodik. Untuk mengonfigurasi fitur ini, ubah strategi sinkronisasi ke`periodic`, dan tentukan a `delay` dalam detik untuk interval tersebut. Untuk informasi selengkapnya, lihat parameter konfigurasi strategi komponen shadow manager.

Pembaruan konfigurasi ini menentukan untuk menyinkronkan bayangan di kedua arah antara AWS IoT Core dan perangkat inti. Jika Anda menggunakan shadow manager v2.2.0 atau yang lebih baru, Anda dapat mengonfigurasi komponen shadow manager untuk menyinkronkan bayangan hanya dalam satu arah. Untuk mengonfigurasi fitur ini, ubah sinkronisasi `direction` ke `deviceToCloud` atau`cloudToDevice`. Untuk informasi selengkapnya, lihat parameter konfigurasi arah komponen shadow manager.

```
{
  "strategy": {
    "type": "realTime"
  },
  "synchronize": {
    "coreThing": {
      "classic": true,
      "namedShadows": [
        "MyCoreShadow"
      ]
    },
    "shadowDocuments": [
      {
        "thingName": "MyDevice1",
        "classic": false,
        "namedShadows": [
          "MyShadowA",
          "MyShadowB"
        ]
      },
      {
        "thingName": "MyDevice2",
        "classic": true,
        "namedShadows": [ ]
      }
    ],
    "direction": "betweenDeviceAndCloud"
  }
```

```
}
```

Sinkronkan bayangan lokal

Saat perangkat inti Greengrass terhubung ke AWS IoT cloud, pengelola bayangan melakukan tugas berikut untuk bayangan yang Anda tentukan dalam konfigurasi komponen. Perilaku tergantung pada opsi konfigurasi arah sinkronisasi bayangan yang Anda tentukan. Secara default, shadow manager menggunakan `betweenDeviceAndCloud` opsi untuk menyinkronkan bayangan di kedua arah. Jika Anda menggunakan shadow manager v2.2.0 atau yang lebih baru, Anda dapat mengonfigurasi perangkat inti untuk menyinkronkan bayangan hanya dalam satu arah, yang bisa `cloudToDevice` atau. `deviceToCloud`

- Jika konfigurasi arah sinkronisasi bayangan adalah `betweenDeviceAndCloud` atau`cloudToDevice`, manajer bayangan mengambil informasi status yang dilaporkan dari dokumen bayangan awan diAWS IoT Core. Kemudian, ia memperbarui dokumen bayangan yang disimpan secara lokal untuk menyinkronkan status perangkat.
- Jika konfigurasi arah sinkronisasi bayangan adalah `betweenDeviceAndCloud` atau`deviceToCloud`, manajer bayangan menerbitkan status perangkat saat ini ke dokumen bayangan awan.

Bayangan menggabungkan perilaku konflik

Dalam beberapa kasus, seperti ketika perangkat inti terputus dari internet, bayangan mungkin berubah di layanan bayangan lokal dan di AWS IoT cloud sebelum manajer bayangan menyinkronkan perubahan. Akibatnya, status yang diinginkan dan dilaporkan berbeda antara layanan bayangan lokal dan AWS IoT cloud

Ketika manajer bayangan menyinkronkan bayangan, itu menggabungkan perubahan sesuai dengan perilaku berikut:

- Jika Anda menggunakan versi pengelola bayangan lebih awal dari v2.2.0, atau saat Anda menentukan arah sinkronisasi `betweenDeviceAndCloud` bayangan, perilaku berikut akan berlaku:
  - Ketika ada konflik gabungan dalam keadaan bayangan yang diinginkan, pengelola bayangan menimpa bagian yang bertentangan dari dokumen bayangan lokal dengan nilai dari cloud. AWS IoT

  - Ketika ada konflik gabungan dalam status bayangan yang dilaporkan, manajer bayangan menimpa bagian bayangan yang bertentangan di AWS IoT cloud dengan nilai dari dokumen bayangan lokal.
- Saat Anda menentukan arah sinkronisasi `deviceToCloud` bayangan, pengelola bayangan akan menimpa bagian bayangan yang bertentangan di AWS IoT cloud dengan nilai dari dokumen bayangan lokal.
- Saat Anda menentukan arah sinkronisasi `cloudToDevice` bayangan, pengelola bayangan akan menimpa bagian yang bertentangan dari dokumen bayangan lokal dengan nilai dari cloud. AWS IoT

Kelola aliran data di perangkat inti Greengrass

Manajer pengaliran AWS IoT Greengrass membuat transfer data IoT volume tinggi ke AWS Cloud lebih efisien dan andal. Stream manager memproses aliran data di inti AWS IoT Greengrass sebelum mengekspornya ke AWS Cloud. Stream manager terintegrasi dengan skenario edge umum, seperti inferensi machine learning (ML), di mana perangkat inti AWS IoT Greengrass memproses dan menganalisis data sebelum mengekspor data tersebut ke AWS Cloud atau tujuan penyimpanan lokal.

Stream manager menyediakan antarmuka umum untuk menyederhanakan pengembangan komponen kustom sehingga Anda tidak perlu membangun fungsi manajemen aliran kustom. Komponen Anda dapat menggunakan mekanisme standar untuk memproses aliran volume tinggi dan mengelola kebijakan penyimpanan data lokal. Anda dapat menentukan kebijakan untuk jenis penyimpanan, ukuran, dan retensi data untuk setiap aliran guna mengontrol cara stream manager memproses dan mengekspor data.

Stream manager bekerja di lingkungan dengan konektivitas yang terputus-putus atau terbatas. Anda dapat menentukan penggunaan bandwidth, perilaku batas waktu, dan cara inti AWS IoT Greengrass menangani data aliran ketika terhubung atau terputus. Anda juga dapat menetapkan prioritas untuk mengontrol urutan di mana inti AWS IoT Greengrass mengekspor aliran ke AWS Cloud. Hal ini memungkinkan Anda untuk menangani data penting lebih cepat dari data lainnya.

Anda dapat mengonfigurasi stream manager untuk mengekspor data secara otomatis ke AWS Cloud untuk disimpan atau diproses dan dianalisis lebih lanjut. Stream manager mendukung ekspor ke tujuan AWS Cloud berikut:

- Saluran di 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 selengkapnya, lihat Apa itu AWS IoT Analytics? dalam Panduan Pengguna AWS IoT Analytics.
- Pengaliran di Amazon Kinesis Data Streams Anda dapat menggunakan Kinesis Data Streams untuk mengumpulkan data volume tinggi dan memuatnya ke gudang data atau cluster. MapReduce Untuk informasi selengkapnya, lihat Apa yang dimaksud dengan Amazon Kinesis Data Streams? dalam Panduan Developer Amazon Kinesis Data Streams.
- Properti aset di AWS IoT SiteWise. AWS IoT SiteWise memungkinkan Anda mengumpulkan, mengatur, dan menganalisis data dari peralatan industri dalam skala besar. Untuk informasi selengkapnya, lihat Apa itu AWS IoT SiteWise? dalam Panduan Pengguna AWS IoT SiteWise.

- Objek di Amazon Simple Storage Service Amazon S3. Sebagai contoh, Anda dapat menggunakan Amazon S3 untuk menyimpan dan mengambil sejumlah besar data. Untuk informasi selengkapnya, lihat Apa itu Amazon S3 dalam Panduan Developer Amazon Simple Storage Service.

Alur kerja manajemen aliran

Aplikasi IoT Anda berinteraksi dengan stream manager melalui SDK Stream Manager.

Dalam alur kerja sederhana, komponen pada inti AWS IoT Greengrass mengonsumsi data IoT, seperti suhu berserial waktu dan metrik tekanan. Komponen tersebut mungkin memfilter atau mengompresi data, dan kemudian memanggil SDK Manajer Pengaliran untuk menuliskan data ke pengaliran di manajer pengaliran. Stream manager dapat mengekspor aliran ke AWS Cloud secara otomatis berdasarkan kebijakan yang Anda tetapkan untuk aliran tersebut. Komponen juga dapat mengirim data langsung ke basis data lokal atau repositori penyimpanan.

Aplikasi IoT Anda dapat mencakup beberapa komponen kustom yang membaca atau menulis ke pengaliran. Komponen-komponen ini dapat membaca dan menulis ke pengaliran untuk memfilter, mengumpulkan, dan menganalisis data pada perangkat inti AWS IoT Greengrass. Hal ini memungkinkan untuk menanggapi dengan cepat peristiwa lokal dan mengambil informasi berharga sebelum data berpindah dari inti ke AWS Cloud atau tujuan lokal.

Untuk memulai, deploy komponen manajer pengaliran ke perangkat inti AWS IoT Greengrass. Dalam deployment, konfigurasi parameter komponen stream manager untuk menentukan pengaturan yang berlaku untuk semua aliran pada perangkat inti Greengrass. Gunakan parameter ini untuk mengontrol cara stream manager menyimpanmem, proses, dan mengekspor aliran berdasarkan kebutuhan bisnis dan kendala lingkungan Anda.

Setelah Anda mengonfigurasi manajer pengaliran, Anda dapat membuat dan men-deploy aplikasi IoT Anda. Hal ini biasanya merupakan komponen kustom yang menggunakan `StreamManagerClient` di SDK Stream Manager untuk membuat dan berinteraksi dengan aliran. Saat membuat aliran, Anda dapat menentukan kebijakan per aliran, seperti tujuan ekspor, prioritas, dan kegigihan.

Persyaratan

Persyaratan berikut berlaku untuk penggunaan stream manager:

- Manajer pengaliran memerlukan minimal 70 MB RAM selain perangkat lunak inti AWS IoT Greengrass. Kebutuhan memori total Anda tergantung pada beban kerja Anda.

- Komponen AWS IoT Greengrass harus menggunakan SDK Manajer Pengaliran untuk berinteraksi dengan manajer pengaliran. SDK Stream Manager tersedia dalam bahasa berikut:
  - SDK for Java Stream Manager (v1.1.0 atau yang lebih baru)
  - SDK for Node.js Stream Manager (v1.1.0 atau yang lebih baru)
  - SDK for Python Stream Manager (v1.1.0 atau yang lebih baru)
- Komponen AWS IoT Greengrass harus menentukan komponen manajer pengaliran (`aws.greengrass.StreamManager`) sebagai dependensi dalam resepnya untuk menggunakan manajer pengaliran.

  > **Note**
  >
  > Jika Anda menggunakan pengelola aliran untuk mengekspor data ke cloud, Anda tidak dapat memutakhirkan versi 2.0.7 komponen pengelola aliran ke versi antara v2.0.8 dan v2.0.11. Jika Anda menerapkan pengelola aliran untuk pertama kalinya, kami sangat menyarankan agar Anda menerapkan versi terbaru komponen pengelola aliran.

- Jika Anda menentukan tujuan ekspor AWS Cloud untuk suatu aliran, Anda harus membuat target ekspor dan memberikan izin akses di peran perangkat Greengrass. Tergantung pada tujuan, persyaratan lain mungkin juga berlaku. Untuk informasi selengkapnya, lihat:
  - the section called “Saluran AWS IoT Analytics”
  - 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 aliran data at rest atau transit di antara komponen-komponen lokal pada perangkat inti.

- Data at rest. Data pengaliran disimpan secara lokal dalam suatu direktori penyimpanan. Untuk keamanan data, AWS IoT Greengrass bergantung pada izin file dan enkripsi full-disk, 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 akan menghapus direktori penyimpanan sebelumnya atau isinya.
- Data saat transit secara lokal. AWS IoT Greengrass tidak mengenkripsi data pengaliran dalam transit lokal antara sumber data, komponen AWS IoT Greengrass, SDK Manajer Pengaliran, dan Manajer Pengaliran.
- Data dalam transit ke AWS Cloud. Aliran data yang diekspor oleh stream manager ke AWS Cloud menggunakan enkripsi klien layanan AWS standar dengan Keamanan Lapisan Pengangkutan (TLS).

Autentikasi Klien

Klien stream manager menggunakan SDK Stream Manager untuk berkomunikasi dengan stream manager. Ketika autentikasi klien diaktifkan, hanya komponen Greengrass yang dapat berinteraksi dengan aliran di stream manager. Ketika autentikasi klien dinonaktifkan, setiap proses yang berjalan pada perangkat inti Greengrass dapat berinteraksi dengan aliran di stream manager. Anda harus menonaktifkan autentikasi hanya jika kasus bisnis Anda memerlukannya.

Anda menggunakan parameter STREAM_MANAGER_AUTHENTICATE_CLIENT untuk mengatur mode autentikasi klien. Anda dapat mengonfigurasikan parameter ini saat Anda men-deploy komponen stream manager tersebut ke perangkat inti Anda.

	Diaktifkan	Dinonaktifkan
Nilai parameter	`true` (default dan disarankan)	`false`
Klien yang diizinkan	Komponen Greengrass pada perangkat inti	Komponen Greengrass pada perangkat inti Proses lain yang berjalan di perangkat inti Greengrass

Lihat juga

- the section called “Konfigurasikan manajer pengaliran”
- the section called “Gunakan StreamManagerClient untuk bekerja dengan aliran”
- the section called “Ekspor konfigurasi untuk tujuan cloud yang didukung”

Buat komponen kustom yang menggunakan stream manager

Gunakan manajer pengaliran pada komponen Greengrass kustom untuk menyimpan, memproses, dan mengekspor data perangkat IoT. Gunakan prosedur dan contoh di bagian ini untuk membuat resep komponen, artefak, dan aplikasi yang bekerja dengan stream manager. Untuk informasi lebih lanjut tentang cara mengembangkan dan menguji komponen, lihat Buat AWS IoT Greengrass komponen.

Topik

- Tentukan resep komponen yang menggunakan stream manager
- Hubungkan ke manajer pengaliran dalam kode aplikasi

Tentukan resep komponen yang menggunakan stream manager

Untuk menggunakan stream manager dalam komponen kustom, Anda harus menentukan komponen `aws.greengrass.StreamManager` sebagai dependensi. Anda juga harus menyediakan SDK Stream Manager. Selesaikan tugas berikut untuk mengunduh dan menggunakan SDK Manajer Pengaliran dalam bahasa pilihan Anda.

Gunakan SDK for Java Stream Manager

SDK for Java Stream Manager tersedia sebagai file JAR yang dapat Anda gunakan untuk mengompilasi komponen Anda. Kemudian, Anda dapat membuat JAR aplikasi yang mencakup SDK Stream Manager, menentukan JAR aplikasi sebagai artefak komponen, dan menjalankan JAR aplikasi dalam siklus hidup komponen.

Untuk menggunakan SDK for Java Stream Manager

1. Unduh file JAR SDK for Java Stream Manager.
2. Lakukan salah satu dari hal berikut untuk membuat artefak komponen dari aplikasi Java Anda dan file JAR SDK Stream Manager:

- Bangun komponen Anda sebagai file JAR yang mencakup JAR SDK Stream Manager, dan jalankan file JAR ini dalam resep komponen Anda.
- Tentukan JAR SDK Stream Manager sebagai artefak komponen. Tambahkan artefak itu ke classpath ketika Anda menjalankan aplikasi Anda dalam resep komponen Anda.

Resep komponen Anda mungkin terlihat serupa dengan contoh berikut ini.
Komponen ini menjalankan versi modifikasi dari contoh StreamManagerS3.java, yang `StreamManagerS3.jar` menyertakan JAR SDK Stream Manager.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.StreamManagerS3Java",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Uses stream manager to upload a file to an S3
 bucket.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.StreamManager": {
      "VersionRequirement": "^2.0.0"
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "java -jar {artifacts:path}/StreamManagerS3.jar"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Java/1.0.0/StreamManagerS3.jar"
        }
      ]
    }
  ]
}
```

YAML

```
---
```

```
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.StreamManagerS3Java
ComponentVersion: 1.0.0
ComponentDescription: Uses stream manager to upload a file to an S3 bucket.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.StreamManager:
    VersionRequirement: "^2.0.0"
Manifests:
  - Lifecycle:
      run: java -jar {artifacts:path}/StreamManagerS3.jar
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Java/1.0.0/StreamManagerS3.jar
```

Untuk informasi lebih lanjut tentang cara mengembangkan dan menguji komponen, lihat Buat AWS IoT Greengrass komponen.

Gunakan SDK for Python Stream Manager

SDK for Python Stream Manager tersedia sebagai kode sumber yang dapat Anda sertakan dalam komponen Anda. Buat file ZIP dari SDK Stream Manager, tentukan file ZIP sebagai artefak komponen, dan instal persyaratan SDK dalam siklus hidup komponen.

Untuk menggunakan SDK for Python Stream Manager

1. Kloning atau unduh repositori aws-greengrass-stream-manager-sdk-python.

   ```
   git clone git@github.com:aws-greengrass/aws-greengrass-stream-manager-sdk-
   python.git
   ```

2. Buat file ZIP yang berisi folder `stream_manager`, yang berisi kode sumber SDK for Python Manajer Pengaliran. Anda dapat menyediakan file ZIP ini sebagai artefak komponen yang dibuka ritsletingnya oleh perangkat lunak AWS IoT Greengrass Core saat menginstal komponen Anda. Lakukan hal-hal berikut:

   a. Buka folder yang berisi repositori yang Anda kloning atau download pada langkah sebelumnya.

```
cd aws-greengrass-stream-manager-sdk-python
```

b. Zip folder `stream_manager` ke dalam file ZIP bernama `stream_manager_sdk.zip`.

Linux or Unix

```
zip -rv stream_manager_sdk.zip stream_manager
```

Windows Command Prompt (CMD)

```
tar -acvf stream_manager_sdk.zip stream_manager
```

PowerShell

```
Compress-Archive stream_manager stream_manager_sdk.zip
```

c. Verifikasi bahwa file `stream_manager_sdk.zip` berisi folder `stream_manager` dan isinya. Jalankan perintah berikut untuk menampilkan konten file ZIP.

Linux or Unix

```
unzip -l stream_manager_sdk.zip
```

Windows Command Prompt (CMD)

```
tar -tf stream_manager_sdk.zip
```

Outputnya akan terlihat serupa dengan yang berikut ini:

```
Archive:  aws-greengrass-stream-manager-sdk-python/stream_manager.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  02-24-2021 20:45   stream_manager/
      913  02-24-2021 20:45   stream_manager/__init__.py
     9719  02-24-2021 20:45   stream_manager/utilinternal.py
     1412  02-24-2021 20:45   stream_manager/exceptions.py
     1004  02-24-2021 20:45   stream_manager/util.py
        0  02-24-2021 20:45   stream_manager/data/
   254463  02-24-2021 20:45   stream_manager/data/__init__.py
```

```
    26515  02-24-2021 20:45   stream_manager/streammanagerclient.py
---------                     -------
   294026                     8 files
```

3. Salin artefak SDK Stream Manager ke folder artefak komponen Anda. Selain file ZIP SDK Stream Manager, komponen Anda menggunakan `requirements.txt` untuk menginstal dependensi SDK Stream Manager. Ganti *`~/greengrass-components`* dengan path ke folder yang akan digunakan untuk pengembangan lokal.

   Linux or Unix

   ```
   cp {stream_manager_sdk.zip,requirements.txt} ~/greengrass-components/artifacts/
   com.example.StreamManagerS3Python/1.0.0/
   ```

   Windows Command Prompt (CMD)

   ```
   robocopy . %USERPROFILE%\greengrass-components\artifacts
   \com.example.StreamManagerS3Python\1.0.0 stream_manager_sdk.zip
   robocopy . %USERPROFILE%\greengrass-components\artifacts
   \com.example.StreamManagerS3Python\1.0.0 requirements.txt
   ```

   PowerShell

   ```
   cp .\stream_manager_sdk.zip,.\requirements.txt ~\greengrass-components\artifacts
   \com.example.StreamManagerS3Python\1.0.0\
   ```

4. Buat resep komponen Anda. Dalam resep tersebut, lakukan hal berikut:

   a. Tentukan `stream_manager_sdk.zip` dan `requirements.txt` sebagai artefak.

   b. Tentukan aplikasi Python Anda sebagai artefak.

   c. Dalam instal siklus hidup, instal persyaratan SDK Stream Manager dari `requirements.txt`.

   d. Dalam jalankan siklus hidup, tambahkan SDK Stream Manager ke `PYTHONPATH`, dan jalankan aplikasi Python Anda.

   Resep komponen Anda mungkin terlihat serupa dengan contoh berikut ini. Komponen ini menjalankan contoh stream_manager_s3.py.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.StreamManagerS3Python",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Uses stream manager to upload a file to an S3
 bucket.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.StreamManager": {
      "VersionRequirement": "^2.0.0"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "pip3 install --user -r {artifacts:path}/requirements.txt",
        "run": "export PYTHONPATH=$PYTHONPATH:{artifacts:decompressedPath}/
stream_manager_sdk; python3 {artifacts:path}/stream_manager_s3.py"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_sdk.zip",
          "Unarchive": "ZIP"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_s3.py"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/requirements.txt"
        }
      ]
    },
    {
      "Platform": {
        "os": "windows"
```

```
      },
      "Lifecycle": {
        "install": "pip3 install --user -r {artifacts:path}/requirements.txt",
        "run": "set \"PYTHONPATH=%PYTHONPATH%;{artifacts:decompressedPath}/
stream_manager_sdk\" & py -3 {artifacts:path}/stream_manager_s3.py"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_sdk.zip",
          "Unarchive": "ZIP"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_s3.py"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/requirements.txt"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.StreamManagerS3Python
ComponentVersion: 1.0.0
ComponentDescription: Uses stream manager to upload a file to an S3 bucket.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.StreamManager:
    VersionRequirement: "^2.0.0"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: pip3 install --user -r {artifacts:path}/requirements.txt
      run: |
```

```
        export PYTHONPATH=$PYTHONPATH:{artifacts:decompressedPath}/
stream_manager_sdk
        python3 {artifacts:path}/stream_manager_s3.py
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_sdk.zip
        Unarchive: ZIP
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_s3.py
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/requirements.txt
  - Platform:
      os: windows
    Lifecycle:
      install: pip3 install --user -r {artifacts:path}/requirements.txt
      run: |
        set "PYTHONPATH=%PYTHONPATH%;{artifacts:decompressedPath}/
stream_manager_sdk"
        py -3 {artifacts:path}/stream_manager_s3.py
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_sdk.zip
        Unarchive: ZIP
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_s3.py
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/requirements.txt
```

Untuk informasi lebih lanjut tentang cara mengembangkan dan menguji komponen, lihat Buat AWS IoT Greengrass komponen.

Menggunakan Stream Manager SDK untuk JavaScript

Stream Manager SDK untuk JavaScript tersedia sebagai kode sumber yang dapat Anda sertakan dalam komponen Anda. Buat file ZIP SDK Stream Manager, tentukan file ZIP sebagai artefak komponen, dan instal SDK dalam siklus hidup komponen.

Untuk menggunakan Stream Manager SDK untuk JavaScript

1. Kloning atau unduh repositori aws-greengrass-stream-manager-sdk-js.

```
git clone git@github.com:aws-greengrass/aws-greengrass-stream-manager-sdk-js.git
```

2. Buat file ZIP yang berisi `aws-greengrass-stream-manager-sdk` folder, yang berisi kode sumber SDK Stream Manager untuk JavaScript. Anda dapat menyediakan file ZIP ini sebagai artefak komponen yang dibuka ritsletingnya oleh perangkat lunak AWS IoT Greengrass Core saat menginstal komponen Anda. Lakukan hal-hal berikut:

   a. Buka folder yang berisi repositori yang Anda kloning atau download pada langkah sebelumnya.

   ```
   cd aws-greengrass-stream-manager-sdk-js
   ```

   b. Zip folder `aws-greengrass-stream-manager-sdk` ke dalam file ZIP bernama `stream-manager-sdk.zip`.

   Linux or Unix

   ```
   zip -rv stream-manager-sdk.zip aws-greengrass-stream-manager-sdk
   ```

   Windows Command Prompt (CMD)

   ```
   tar -acvf stream-manager-sdk.zip aws-greengrass-stream-manager-sdk
   ```

   PowerShell

   ```
   Compress-Archive aws-greengrass-stream-manager-sdk stream-manager-sdk.zip
   ```

   c. Verifikasi bahwa file `stream-manager-sdk.zip` berisi folder `aws-greengrass-stream-manager-sdk` dan isinya. Jalankan perintah berikut untuk menampilkan konten file ZIP.

   Linux or Unix

   ```
   unzip -l stream-manager-sdk.zip
   ```

   Windows Command Prompt (CMD)

   ```
   tar -tf stream-manager-sdk.zip
   ```

Outputnya akan terlihat serupa dengan yang berikut ini:

```
Archive:  stream-manager-sdk.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/
      369  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/package.json
     1017  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/util.js
     8374  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/utilInternal.js
     1937  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/exceptions.js
        0  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/data/
   353343  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/data/index.js
    22599  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/client.js
      216  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/index.js
---------                     -------
   387855                     9 files
```

3. Salin artefak SDK Stream Manager ke folder artefak komponen Anda. Ganti *~/greengrass-components* dengan path ke folder yang akan digunakan untuk pengembangan lokal.

   Linux or Unix

   ```
   cp stream-manager-sdk.zip ~/greengrass-components/artifacts/
   com.example.StreamManagerS3JS/1.0.0/
   ```

   Windows Command Prompt (CMD)

   ```
   robocopy . %USERPROFILE%\greengrass-components\artifacts
   \com.example.StreamManagerS3JS\1.0.0 stream-manager-sdk.zip
   ```

   PowerShell

   ```
   cp .\stream-manager-sdk.zip ~\greengrass-components\artifacts
   \com.example.StreamManagerS3JS\1.0.0\
   ```

4. Buat resep komponen Anda. Dalam resep tersebut, lakukan hal berikut:

   a. Tentukan `stream-manager-sdk.zip` sebagai artefak.

   b. Tentukan JavaScript aplikasi Anda sebagai artefak.

c. Pada instal siklus hidup, instal SDK Manajer Pengaliran dari artefak `stream-manager-sdk.zip`. Perintah `npm install` membuat folder `node_modules` yang berisi SDK Stream Manager dan dependensinya.

d. Dalam siklus hidup run, tambahkan `node_modules` folder ke`NODE_PATH`, dan jalankan aplikasi Anda. JavaScript

Resep komponen Anda mungkin terlihat serupa dengan contoh berikut ini. Komponen ini menjalankan contoh StreamManagerS3.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.StreamManagerS3JS",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Uses stream manager to upload a file to an S3
 bucket.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.StreamManager": {
      "VersionRequirement": "^2.0.0"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "npm install {artifacts:decompressedPath}/stream-manager-sdk/
aws-greengrass-stream-manager-sdk",
        "run": "export NODE_PATH=$NODE_PATH:{work:path}/node_modules; node
 {artifacts:path}/index.js"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/stream-manager-sdk.zip",
          "Unarchive": "ZIP"
        },
        {
```

```
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/index.js"
        }
      ]
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "npm install {artifacts:decompressedPath}/stream-manager-sdk/
aws-greengrass-stream-manager-sdk",
        "run": "set \"NODE_PATH=%NODE_PATH%;{work:path}/node_modules\" & node
 {artifacts:path}/index.js"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/stream-manager-sdk.zip",
          "Unarchive": "ZIP"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/index.js"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.StreamManagerS3JS
ComponentVersion: 1.0.0
ComponentDescription: Uses stream manager to upload a file to an S3 bucket.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.StreamManager:
    VersionRequirement: "^2.0.0"
Manifests:
  - Platform:
```

```
      os: linux
    Lifecycle:
      install: npm install {artifacts:decompressedPath}/stream-manager-sdk/aws-
greengrass-stream-manager-sdk
      run: |
        export NODE_PATH=$NODE_PATH:{work:path}/node_modules
        node {artifacts:path}/index.js
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/stream-manager-sdk.zip
        Unarchive: ZIP
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/index.js
  - Platform:
      os: windows
    Lifecycle:
      install: npm install {artifacts:decompressedPath}/stream-manager-sdk/aws-
greengrass-stream-manager-sdk
      run: |
        set "NODE_PATH=%NODE_PATH%;{work:path}/node_modules"
        node {artifacts:path}/index.js
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/stream-manager-sdk.zip
        Unarchive: ZIP
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/index.js
```

Untuk informasi lebih lanjut tentang cara mengembangkan dan menguji komponen, lihat Buat AWS IoT Greengrass komponen.

Hubungkan ke manajer pengaliran dalam kode aplikasi

Untuk terhubung ke stream manager dalam aplikasi Anda, buat sebuah instans `StreamManagerClient` dari SDK Stream Manager. Klien ini terhubung ke komponen stream manager pada port default 8088, atau port yang Anda tentukan. Untuk informasi lebih lanjut tentang cara menggunakan `StreamManagerClient` setelah Anda membuat instans, lihat Gunakan StreamManagerClient untuk bekerja dengan aliran.

Example Contoh: Hubungkan ke manajer pengaliran dengan port default

Java

```
import com.amazonaws.greengrass.streammanager.client.StreamManagerClient;

public class MyStreamManagerComponent {

    void connectToStreamManagerWithDefaultPort() {
        StreamManagerClient client = StreamManagerClientFactory.standard().build();

        // Use the client.
    }
}
```

Python

```
from stream_manager import (
    StreamManagerClient
)

def connect_to_stream_manager_with_default_port():
    client = StreamManagerClient()

    # Use the client.
```

JavaScript

```
const {
    StreamManagerClient
} = require('aws-greengrass-stream-manager-sdk');

function connectToStreamManagerWithDefaultPort() {
    const client = new StreamManagerClient();

    // Use the client.
}
```

Example Contoh: Hubungkan ke manajer pengaliran dengan port non-default

Jika Anda mengonfigurasi stream manager dengan port selain default, Anda harus menggunakan komunikasi antar proses untuk mengambil port dari konfigurasi komponen.

> **Note**
>
> Parameter konfigurasi `port` berisi nilai yang Anda tentukan di `STREAM_MANAGER_SERVER_PORT` saat Anda menggunakan manajer pengaliran.

Java

```
void connectToStreamManagerWithCustomPort() {
    EventStreamRPCConnection eventStreamRpcConnection =
 IPCUtils.getEventStreamRpcConnection();
    GreengrassCoreIPCClient greengrassCoreIPCClient = new
 GreengrassCoreIPCClient(eventStreamRpcConnection);
    List<String> keyPath = new ArrayList<>();
    keyPath.add("port");

    GetConfigurationRequest request = new GetConfigurationRequest();
    request.setComponentName("aws.greengrass.StreamManager");
    request.setKeyPath(keyPath);
    GetConfigurationResponse response =
            greengrassCoreIPCClient.getConfiguration(request,
 Optional.empty()).getResponse().get();
    String port = response.getValue().get("port").toString();
    System.out.print("Stream Manager is running on port: " + port);

    final StreamManagerClientConfig config = StreamManagerClientConfig.builder()

 .serverInfo(StreamManagerServerInfo.builder().port(Integer.parseInt(port)).build()).build()

    StreamManagerClient client =
 StreamManagerClientFactory.standard().withClientConfig(config).build();

    // Use the client.
}
```

Python

```
import awsiot.greengrasscoreipc
from awsiot.greengrasscoreipc.model import (
    GetConfigurationRequest
)
from stream_manager import (
    StreamManagerClient
)

TIMEOUT = 10

def connect_to_stream_manager_with_custom_port():
    # Use IPC to get the port from the stream manager component configuration.
    ipc_client = awsiot.greengrasscoreipc.connect()
    request = GetConfigurationRequest()
    request.component_name = "aws.greengrass.StreamManager"
    request.key_path = ["port"]
    operation = ipc_client.new_get_configuration()
    operation.activate(request)
    future_response = operation.get_response()
    response = future_response.result(TIMEOUT)
    stream_manager_port = str(response.value["port"])

    # Use port to create a stream manager client.
    stream_client = StreamManagerClient(port=stream_manager_port)

    # Use the client.
```

Gunakan StreamManagerClient untuk bekerja dengan aliran

Komponen Greengrass yang ditentukan oleh pengguna yang berjalan pada perangkat inti Greengrass dapat menggunakan objek `StreamManagerClient` di SDK Stream Manager untuk membuat aliran di stream manager dan kemudian berinteraksi dengan aliran tersebut. Ketika komponen menciptakan pengaliran, komponen tersebut menentukan tujuan AWS Cloud, prioritas, serta kebijakan ekspor dan retensi data lainnya untuk pengaliran tersebut. Untuk mengirim data ke stream manager, komponen menambahkan data ke aliran. Jika tujuan ekspor telah ditentukan untuk aliran tersebut, stream manager akan mengekspor aliran tersebut secara otomatis.

> **Note**
>
> Biasanya, klien stream manager adalah komponen Greengrass yang ditetapkan pengguna. Jika kasus bisnis Anda memerlukannya, Anda juga dapat mengizinkan proses non-komponen yang berjalan pada inti Greengrass (misalnya, kontainer Docker) untuk berinteraksi dengan stream manager. Untuk informasi selengkapnya, lihat the section called “Autentikasi Klien”.

Cuplikan dalam topik ini menunjukkan cara klien memanggil metode `StreamManagerClient` untuk bekerja dengan aliran. Untuk detail implementasi tentang metode dan argumennya, gunakan tautan ke referensi SDK terdaftar setelah setiap potongan.

Jika Anda menggunakan stream manager dalam fungsi Lambda, fungsi Lambda Anda harus memberi contoh `StreamManagerClient` di luar fungsi handler. Jika dipakai dalam handler, fungsi tersebut akan membuat `client` dan koneksi ke manajer 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 “Buat aliran 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”

Buat aliran pesan

Untuk membuat pengaliran, komponen Greengrass yang ditetapkan pengguna akan memanggil metode buat dan melewati objek `MessageStreamDefinition`. Objek ini menentukan nama unik

untuk aliran tersebut dan menentukan bagaimana stream manager harus menangani data baru ketika ukuran aliran 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 “Ekspor konfigurasi untuk tujuan 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.
- T ime-to-live (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:

- MessageStreamDefinitiondi Java SDK
- MessageStreamDefinitiondi Node.js SDK
- MessageStreamDefinitiondi Python SDK

> **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.

Setelah stream dibuat, komponen Greengrass Anda dapat menambahkan pesan ke aliran tersebut untuk mengirim data untuk ekspor dan membaca pesan dari aliran tersebut untuk pemrosesan lokal. Jumlah aliran 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:

- Versi SDK Stream Manager minimum: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Contoh-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 Python SDK: create_message_stream | MessageStreamDefinition

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    client.createMessageStream(
            new MessageStreamDefinition()
                    .withName("StreamName") // Required.
                    .withMaxSize(268435456L)    // Default is 256 MB.
                    .withStreamSegmentSize(16777216L)    // Default is 16 MB.
                    .withTimeToLiveMillis(null)    // By default, no TTL is enabled.
                    .withStrategyOnFull(StrategyOnFull.OverwriteOldestData)    //
 Required.
                    .withPersistence(Persistence.File)    // Default is File.
                    .withFlushOnWrite(false)    // Default is false.
                    .withExportDefinition(    // Optional. Choose where/how the
 stream is exported to the AWS Cloud.
                            new ExportDefinition()
                                    .withKinesis(null)
                                    .withIotAnalytics(null)
                                    .withIotSitewise(null)
                                    .withS3(null)
                    )

    );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: | createMessageStreamMessageStreamDefinition

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)
            .withS3(null)
        )
    );
  } catch (e) {
    // Properly handle errors.
  }
});
client.onError((err) => {
  // Properly handle connection errors.
  // This is called only when the connection to the StreamManager server fails.
});
```

Referensi Node.js SDK: | createMessageStreamMessageStreamDefinition

Untuk informasi lebih lanjut tentang konfigurasi tujuan ekspor, lihatthe section called “Ekspor konfigurasi untuk tujuan cloud yang didukung”.

Tambahkan pesan

Untuk mengirim data ke stream manager untuk ekspor, komponen Greengrass Anda akan menambahkan data ke target aliran. Tujuan ekspor menentukan jenis data yang akan lolos ke metode ini.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Versi SDK Stream Manager minimum: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Contoh-contoh

AWS IoT Analytics atau tujuan ekspor Kinesis Data Streams

Potongan berikut menambahkan pesan ke aliran bernama `StreamName`. Untuk AWS IoT Analytics atau tujuan Kinesis Data Streams, komponen Greengrass Anda menambahkan sekumpulan data.

Potongan ini memiliki persyaratan sebagai berikut:

- Versi SDK Stream Manager minimum: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Python

```
client = StreamManagerClient()

try:
    sequence_number = client.append_message(stream_name="StreamName",
 data=b'Arbitrary bytes data')
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi SDK Python: append_message

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    long sequenceNumber = client.appendMessage("StreamName", "Arbitrary byte
 array".getBytes());
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

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

Potongan berikut menambahkan pesan ke aliran bernama `StreamName`. Untuk tujuan AWS IoT SiteWise, komponen Greengrass Anda akan menambahkan objek `PutAssetPropertyValueEntry` berserial. Untuk informasi selengkapnya, lihat the section called "Ekspor 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 dalam Referensi API AWS IoT SiteWise.

Potongan ini memiliki persyaratan sebagai berikut:

- Versi SDK Stream Manager minimum: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Python

```
client = StreamManagerClient()

try:
    # SiteWise requires unique timestamps in all messages and also needs timestamps
 not earlier
```

```
    # than 10 minutes in the past. Add some randomness to time and offset.

    # Note: To create a new asset property data, you should use the classes defined
 in the
    # greengrasssdk.stream_manager module.

    time_in_nanos = TimeInNanos(
        time_in_seconds=calendar.timegm(time.gmtime()) - random.randint(0, 60),
 offset_in_nanos=random.randint(0, 10000)
    )
    variant = Variant(double_value=random.random())
    asset = [AssetPropertyValue(value=variant, quality=Quality.GOOD,
 timestamp=time_in_nanos)]
    putAssetPropertyValueEntry =
 PutAssetPropertyValueEntry(entry_id=str(uuid.uuid4()),
 property_alias="PropertyAlias", property_values=asset)
    sequence_number = client.append_message(stream_name="StreamName",
 Util.validate_and_serialize_to_json_bytes(putAssetPropertyValueEntry))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Referensi Python SDK: append_message | PutAssetPropertyValueEntry

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    Random rand = new Random();
    // Note: To create a new asset property data, you should use the classes defined
 in the
    // com.amazonaws.greengrass.streammanager.model.sitewise package.
    List<AssetPropertyValue> entries = new ArrayList<>() ;

    // IoTSiteWise requires unique timestamps in all messages and also needs
 timestamps not earlier
    // than 10 minutes in the past. Add some randomness to time and offset.
    final int maxTimeRandomness = 60;
    final int maxOffsetRandomness = 10000;
    double randomValue = rand.nextDouble();
    TimeInNanos timestamp = new TimeInNanos()
```

```
            .withTimeInSeconds(Instant.now().getEpochSecond() -
 rand.nextInt(maxTimeRandomness))
            .withOffsetInNanos((long) (rand.nextInt(maxOffsetRandomness)));
    AssetPropertyValue entry = new AssetPropertyValue()
            .withValue(new Variant().withDoubleValue(randomValue))
            .withQuality(Quality.GOOD)
            .withTimestamp(timestamp);
    entries.add(entry);

    PutAssetPropertyValueEntry putAssetPropertyValueEntry = new
 PutAssetPropertyValueEntry()
            .withEntryId(UUID.randomUUID().toString())
            .withPropertyAlias("PropertyAlias")
            .withPropertyValues(entries);
    long sequenceNumber = client.appendMessage("StreamName",
 ValidateAndSerialize.validateAndSerializeToJsonBytes(putAssetPropertyValueEntry));
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi Java SDK: AppendMessage | PutAssetPropertyValueEntry

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const maxTimeRandomness = 60;
        const maxOffsetRandomness = 10000;
        const randomValue = Math.random();
        // Note: To create a new asset property data, you should use the classes
 defined in the
        // aws-greengrass-core-sdk StreamManager module.
        const timestamp = new TimeInNanos()
            .withTimeInSeconds(Math.round(Date.now() / 1000) -
 Math.floor(Math.random() - maxTimeRandomness))
            .withOffsetInNanos(Math.floor(Math.random() * maxOffsetRandomness));
        const entry = new AssetPropertyValue()
            .withValue(new Variant().withDoubleValue(randomValue))
            .withQuality(Quality.GOOD)
            .withTimestamp(timestamp);

        const putAssetPropertyValueEntry =    new PutAssetPropertyValueEntry()
            .withEntryId(`${ENTRY_ID_PREFIX}${i}`)
```

```
            .withPropertyAlias("PropertyAlias")
            .withPropertyValues([entry]);
        const sequenceNumber = await client.appendMessage("StreamName",
 util.validateAndSerializeToJsonBytes(putAssetPropertyValueEntry));
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi Node.js SDK: AppendMessage | PutAssetPropertyValueEntry

Tujuan ekspor Amazon S3

Potongan berikut menambahkan tugas ekspor ke aliran bernama `StreamName`. Untuk tujuan Amazon S3, komponen Greengrass Anda menambahkan serial objek `S3ExportTaskDefinition` yang berisi informasi tentang file input sumber dan target objek Amazon S3. Jika objek tertentu tidak ada, Stream Manager akan membuatnya untuk Anda. Untuk informasi selengkapnya, lihat the section called “Mengekspor ke Amazon S3”.

Potongan ini memiliki persyaratan sebagai berikut:

- Versi SDK Stream Manager minimum: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Python

```
client = StreamManagerClient()

try:
    # Append an Amazon S3 Task definition and print the sequence number.
    s3_export_task_definition = S3ExportTaskDefinition(input_url="URLToFile",
 bucket="BucketName", key="KeyName")
    sequence_number = client.append_message(stream_name="StreamName",
 Util.validate_and_serialize_to_json_bytes(s3_export_task_definition))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
```

```
    pass
    # Properly handle errors.
```

Referensi Python SDK: append_message | S3 ExportTaskDefinition

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    // Append an Amazon S3 export task definition and print the sequence number.
    S3ExportTaskDefinition s3ExportTaskDefinition = new S3ExportTaskDefinition()
        .withBucket("BucketName")
        .withKey("KeyName")
        .withInputUrl("URLToFile");
    long sequenceNumber = client.appendMessage("StreamName",
 ValidateAndSerialize.validateAndSerializeToJsonBytes(s3ExportTaskDefinition));
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi Java SDK: AppendMessage | S3 ExportTaskDefinition

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
     // Append an Amazon S3 export task definition and print the sequence number.
     const taskDefinition = new S3ExportTaskDefinition()
        .withBucket("BucketName")
        .withKey("KeyName")
        .withInputUrl("URLToFile");
        const sequenceNumber = await client.appendMessage("StreamName",
 util.validateAndSerializeToJsonBytes(taskDefinition)));
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi Node.js SDK: AppendMessage | S3 ExportTaskDefinition

Baca pesan

Baca pesan dari suatu aliran.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Versi SDK Stream Manager minimum: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Contoh-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 Python SDK: read_messages | ReadMessagesOptions

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    List<Message> messages = client.readMessages("StreamName",
            // By default, if no options are specified, it tries to read one message
 from the beginning of the stream.
            new ReadMessagesOptions()
                    // Try to read from sequence number 100 or greater. By default
 this is 0.
                    .withDesiredStartSequenceNumber(100L)
                    // Try to read 10 messages. If 10 messages are not available,
 then NotEnoughMessagesException is raised. By default, this is 1.
                    .withMinMessageCount(10L)
                    // Accept up to 100 messages. By default this is 1.
                    .withMaxMessageCount(100L)
                    // Try to wait at most 5 seconds for the min_messsage_count to
 be fulfilled. By default, this is 0, which immediately returns the messages or an
 exception.
                    .withReadTimeoutMillis(Duration.ofSeconds(5L).toMillis())
    );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

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 Node.js SDK: ReadMessages | ReadMessagesOptions

Daftar aliran

Dapatkan daftar aliran di stream manager.

Persyaratan

Komponen ini memiliki persyaratan sebagai berikut:

- Versi SDK Stream Manager minimum: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Contoh-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 =
 StreamManagerClientFactory.standard().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:

- Versi SDK Stream Manager minimum: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Contoh-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 =
 StreamManagerClientFactory.standard().build()) {
    MessageStreamInfo description = client.describeMessageStream("StreamName");
    String lastErrorMessage =
 description.getExportStatuses().get(0).getErrorMessage();
```

```
    if (lastErrorMessage != null && !lastErrorMessage.equals("")) {
        // The last export of export destination 0 failed with some error.
        // Here is the last sequence number that was successfully exported.
        description.getExportStatuses().get(0).getLastExportedSequenceNumber();
    }

    if (description.getStorageStatus().getNewestSequenceNumber() >
            description.getExportStatuses().get(0).getLastExportedSequenceNumber())
 {
        // The end of the stream is ahead of the last exported sequence number.
    }
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi Java SDK: describeMessageStream

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const description = await client.describeMessageStream("StreamName");
        const lastErrorMessage = description.exportStatuses[0].errorMessage;
        if (lastErrorMessage) {
            // The last export of export destination 0 failed with some error.
            // Here is the last sequence number that was successfully exported.
            description.exportStatuses[0].lastExportedSequenceNumber;
        }

        if (description.storageStatus.newestSequenceNumber >
            description.exportStatuses[0].lastExportedSequenceNumber) {
            // The end of the stream is ahead of the last exported sequence number.
        }
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi Node.js SDK: describeMessageStream

Perbarui aliran pesan

Perbarui properti dari aliran yang ada. Anda mungkin ingin memperbarui aliran jika kebutuhan Anda berubah setelah aliran dibuat. Sebagai contoh:

- 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; misalnya, jika tugas ekspor berjalan lama dan Anda ingin membagi data unggahan Anda.

Komponen Greengrass Anda mengikuti proses tingkat tinggi ini untuk memperbarui aliran:

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:

- Versi SDK Stream Manager minimum: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Contoh-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 Python SDK: | updateMessageStreamMessageStreamDefinition

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    MessageStreamInfo messageStreamInfo = client.describeMessageStream(STREAM_NAME);
    // Update the message stream with new values.
    client.updateMessageStream(
        messageStreamInfo.getDefinition()
            .withStrategyOnFull(StrategyOnFull.RejectNewData) // Required. Updating
 Strategy on full to reject new data.
           // Max Size update should be greater than initial Max Size defined in
 Create Message Stream request
            .withMaxSize(536870912L) // Update Max Size to 512 MB.
            .withStreamSegmentSize(33554432L) // Update Segment Size to 32 MB.
            .withFlushOnWrite(true) // Update flush on write to true.
            .withPersistence(Persistence.Memory) // Update the persistence to
 Memory.
            .withTimeToLiveMillis(3600000L)    // Update TTL to 1 hour.
            .withExportDefinition(
                // Optional. Choose where/how the stream is exported to the 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 Node.js SDK: | updateMessageStreamMessageStreamDefinition

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, jelaskan aliran dan kemudian periksa status penyimpanan `MessageStreamInfo` objek yang dikembalikan.
  - 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, jelaskan aliran dan kemudian periksa status penyimpanan `MessageStreamInfo` objek yang dikembalikan.

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:

- Versi SDK Stream Manager minimum: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Contoh-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 Python SDK: deleteMessageStream

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().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.
});
```

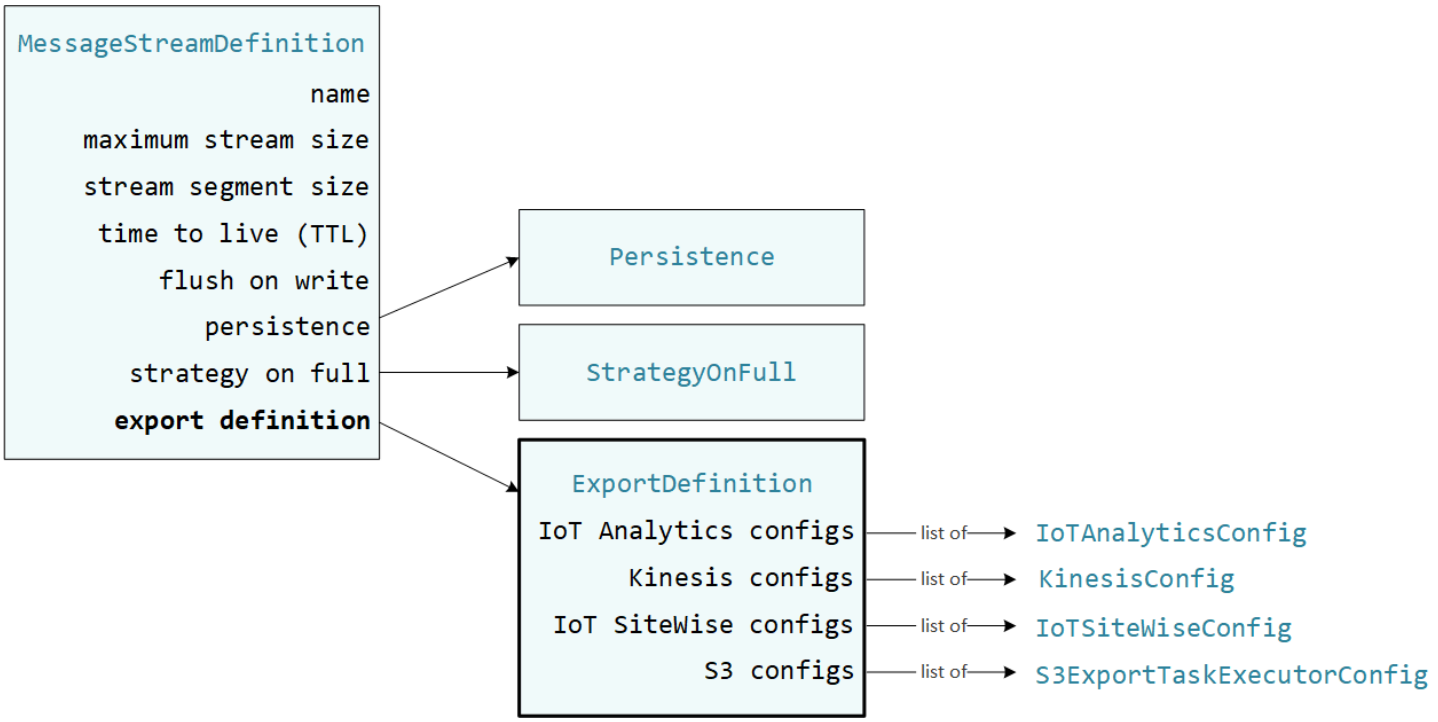
Referensi Node.js SDK: deleteMessageStream

Lihat juga

- Kelola aliran data di perangkat inti Greengrass
- Konfigurasikan manajer pengaliran AWS IoT Greengrass
- Ekspor konfigurasi untuk tujuan AWS Cloud yang didukung
- `StreamManagerClient` dalam referensi SDK Stream Manager:
  - Python
  - Java
  - Node.js

Ekspor konfigurasi untuk tujuan AWS Cloud yang didukung

Komponen Greengrass yang ditentukan pengguna menggunakan `StreamManagerClient` di SDK Stream Manager untuk berinteraksi dengan stream manager. Ketika komponen membuat aliran atau memperbarui aliran, melewati objek `MessageStreamDefinition` yang mewakili properti aliran, termasuk definisi ekspor. Objek `ExportDefinition` berisi konfigurasi ekspor yang ditentukan untuk aliran tersebut. 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

- Saluran AWS IoT Analytics
- Amazon Kinesis data streams
- Properti aset AWS IoT SiteWise
- Objek Amazon S3

Anda bertanggung jawab untuk menjaga sumber daya AWS Cloud ini.

Saluran AWS IoT Analytics

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 selengkapnya, lihat Apa itu AWS IoT Analytics? di Panduan Pengguna AWS IoT Analytics.

Dalam SDK Stream Manager, komponen Greengrass Anda menggunakan `IoTAnalyticsConfig` untuk menentukan konfigurasi ekspor untuk jenis tujuan ini. Untuk informasi lebih lanjut, lihat referensi SDK untuk bahasa target Anda:

- IoT AnalyticsConfig dalam SDK Python
- IoT AnalyticsConfig di Java SDK
- IoT AnalyticsConfig di Node.js SDK

Persyaratan

Tujuan ekspor ini memiliki persyaratan sebagai berikut:

- Targetkan saluran dalam AWS IoT Analytics harus dalam Akun AWS dan Wilayah AWS yang sama sebagai perangkat inti Greengrass.
- Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan harus memungkinkan izin `iotanalytics:BatchPutMessage` untuk menargetkan saluran. Sebagai contoh:

```
{
  "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.

Ekspor ke AWS IoT Analytics

Untuk membuat pengaliran yang diekspor ke AWS IoT Analytics, komponen Greengrass Anda membuat pengaliran dengan definisi ekspor yang mencakup satu atau lebih objek `IoTAnalyticsConfig`. Objek ini menentukan pengaturan ekspor, seperti target saluran, ukuran batch, interval batch, dan prioritas.

Ketika komponen Greengrass Anda menerima data dari perangkat, komponen itu menambahkan pesan yang berisi gumpalan data ke aliran target.

Kemudian, stream manager mengekspor data berdasarkan pengaturan batch dan prioritas yang ditentukan dalam konfigurasi ekspor aliran.

Amazon Kinesis data streams

Stream manager mendukung ekspor otomatis ke Amazon Kinesis Data Streams. Kinesis Data Streams biasanya digunakan untuk mengumpulkan data volume tinggi dan memuatnya ke dalam gudang data atau cluster. MapReduce Untuk informasi selengkapnya, lihat Apa yang dimaksud dengan Amazon Kinesis Data Streams? dalam Panduan Developer Amazon Kinesis.

Dalam SDK Stream Manager, komponen Greengrass Anda menggunakan `KinesisConfig` untuk menentukan konfigurasi ekspor untuk jenis tujuan ini. Untuk informasi lebih lanjut, lihat referensi SDK untuk bahasa target Anda:

- KinesisConfigdi Python SDK

- KinesisConfigdi Java SDK
- KinesisConfigdi Node.js SDK

Persyaratan

Tujuan ekspor ini memiliki persyaratan sebagai berikut:

- Target aliran di Kinesis Data Streams harus berada di Akun AWS dan Wilayah AWS yang sama dengan perangkat inti Greengrass.
- Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan harus memungkinkan izin `kinesis:PutRecords` untuk menargetkan aliran data. Sebagai contoh:

```
{
  "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 diekspor ke Kinesis Data Streams, komponen Greengrass Anda membuat pengaliran dengan definisi ekspor yang mencakup satu atau lebih objek `KinesisConfig`. Objek ini menentukan pengaturan ekspor, seperti target aliran data, ukuran batch, interval batch, dan prioritas.

Ketika komponen Greengrass Anda menerima data dari perangkat, komponen itu menambahkan pesan yang berisi gumpalan data ke aliran target. Kemudian, stream manager mengekspor data berdasarkan pengaturan batch dan prioritas yang ditentukan dalam konfigurasi ekspor aliran.

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 selengkapnya, lihat Apa itu AWS IoT SiteWise? di Panduan Pengguna AWS IoT SiteWise.

Dalam SDK Stream Manager, komponen Greengrass Anda menggunakan `IoTSiteWiseConfig` untuk menentukan konfigurasi ekspor untuk jenis tujuan ini. Untuk informasi lebih lanjut, lihat referensi SDK untuk bahasa target Anda:

- IoT SiteWiseConfig dalam SDK Python
- IoT SiteWiseConfig di Java SDK
- IoT SiteWiseConfig di Node.js SDK

> **Note**
>
> AWSjuga menyediakan AWS IoT SiteWise komponen, yang menawarkan solusi pra-bangun yang dapat Anda gunakan untuk mengalirkan data dari sumber OPC-UA. Untuk informasi selengkapnya, lihat Kolektor IoT SiteWise OPC-UA.

Persyaratan

Tujuan ekspor ini memiliki persyaratan sebagai berikut:

- Target properti aset AWS IoT SiteWise harus berada di Akun AWS dan Wilayah AWS yang sama dengan perangkat inti Greengrass.

> **Note**
>
> Untuk daftar Wilayah AWS yang didukung oleh AWS IoT SiteWise, lihat titik akhir dan kuotaAWS IoT SiteWise dalam Referensi Umum AWS.

- Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan harus memungkinkan izin `iotsitewise:BatchPutAssetPropertyValue` 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 otorisasi di Panduan AWS IoT SiteWise Pengguna.

Ekspor ke AWS IoT SiteWise

Untuk membuat pengaliran yang diekspor ke AWS IoT SiteWise, komponen Greengrass Anda membuat pengaliran dengan definisi ekspor yang mencakup satu atau lebih objek `IoTSiteWiseConfig`. Objek ini menentukan pengaturan ekspor, seperti ukuran batch, interval batch, dan prioritas.

Ketika komponen Greengrass Anda menerima data properti aset dari perangkat, komponen itu menambahkan pesan yang berisi data ke aliran target. Pesan merupakan objek `PutAssetPropertyValueEntry` berseri 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 dalam Referensi API AWS IoT SiteWise.

Kemudian, stream manager mengekspor data berdasarkan pengaturan batch dan prioritas yang ditentukan dalam konfigurasi ekspor aliran.

Anda dapat menyesuaikan pengaturan stream manager dan logika komponen Greengrass untuk merancang strategi ekspor Anda. Sebagai contoh:

- 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 batasan bandwidth, atau meminimalkan biaya, komponen Greengrass Anda dapat mengumpulkan titik data timestamp-quality-value (TQV) yang diterima untuk satu properti aset 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 selengkapnya, lihat Apa itu Amazon S3? dalam Panduan Developer Amazon Simple Storage Service.

Dalam SDK Stream Manager, komponen Greengrass Anda menggunakan `S3ExportTaskExecutorConfig` untuk menentukan konfigurasi ekspor untuk jenis tujuan ini. Untuk informasi lebih lanjut, lihat referensi SDK untuk bahasa target Anda:

- S3 ExportTaskExecutorConfig di Python SDK
- S3 ExportTaskExecutorConfig di Java SDK
- S3 ExportTaskExecutorConfig di Node.js SDK

Persyaratan

Tujuan ekspor ini memiliki persyaratan sebagai berikut:

- Target bucket Amazon S3 harus berada di Akun AWS yang sama dengan perangkat inti Greengrass.
- Jika fungsi Lambda yang berjalan dalam mode container Greengrass menulis file input ke direktori file input, Anda harus me-mount direktori sebagai volume dalam wadah dengan izin tulis. Ini memastikan bahwa file ditulis ke sistem file root dan terlihat oleh komponen manajer aliran, yang berjalan di luar wadah.
- Jika komponen kontainer Docker menulis file input ke direktori file input, Anda harus me-mount direktori sebagai volume dalam wadah dengan izin tulis. Ini memastikan bahwa file ditulis ke sistem file root dan terlihat oleh komponen manajer aliran, yang berjalan di luar wadah.
- Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan harus memungkinkan izin berikut untuk target bucket. Sebagai contoh:

```
{
  "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 aliran yang diekspor ke Amazon S3, komponen Greengrass Anda menggunakan objek `S3ExportTaskExecutorConfig` untuk mengonfigurasi kebijakan ekspor. Kebijakan ini menentukan pengaturan ekspor, seperti ambang batas dan prioritas unggahan multipart. Untuk ekspor Amazon S3, stream manager akan mengunggah data yang dibaca dari file lokal pada perangkat inti. Untuk memulai unggahan, komponen Greengrass Anda menambahkan tugas ekspor ke aliran target. Tugas ekspor berisi informasi tentang file input dan objek Amazon S3 target. Stream manager menjalankan tugas dalam urutan yang ditambahkan ke aliran.

> **Note**
>
> Bucket target harus sudah ada dalam perangkat Akun AWS. Jika objek untuk kunci tertentu tidak ada, stream manager akan membuatnya untuk Anda.

Manajer pengaliran menggunakan properti ambang batas unggahan multipart, pengaturan ukuran bagian minimum, dan ukuran file input untuk menentukan cara 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 menyertakan DateTimeFormatter string Java 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 menuliskan kode yang digunakan oleh aplikasi IoT untuk mengelola siklus hidup data input. Contoh alur kerja berikut menunjukkan bagaimana Anda mungkin menggunakan komponen Greengrass untuk mengelola data ini.

1. Proses lokal menerima data dari perangkat atau periferal, dan kemudian menuliskan data ke file dalam direktori pada perangkat inti. Ini adalah file input untuk stream manager.
2. Komponen Greengrass memindai direktori dan menambahkan tugas ekspor ke aliran target ketika file baru dibuat. Tugas tersebut adalah objek `S3ExportTaskDefinition` JSON berseri yang menentukan URL 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 objek untuk kunci tertentu tidak ada, stream manager akan membuatnya untuk Anda.
4. Komponen Greengrass membaca pesan dari aliran status untuk memantau status ekspor. Setelah tugas ekspor selesai, komponen Greengrass dapat menghapus file input yang sesuai. Untuk informasi selengkapnya, lihat the section called “Pantau tugas ekspor”.

Pantau tugas ekspor

Anda dapat menuliskan kode yang digunakan oleh aplikasi IoT untuk memantau status ekspor Amazon S3 Anda. Komponen Greengrass Anda harus membuat status aliran dan kemudian mengonfigurasi aliran ekspor untuk menuliskan pembaruan status pada status aliran. Aliran status tunggal dapat menerima pembaruan status dari beberapa aliran yang diekspor 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. Sebagai contoh:

- 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 pada konfigurasi ekspor `S3ExportTaskExecutorConfig` aliran. Pengaturan ini memberitahu stream manager untuk menuliskan pesan status tentang tugas ekspor ke status aliran. Di objek `StatusConfig`, tentukan nama status pengaliran dan tingkat verbositasnya. 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 bagaimana komponen Greengrass mungkin menggunakan status aliran untuk memantau status ekspor.

1. Seperti yang dijelaskan dalam alur kerja sebelumnya, sebuah komponen Greengrass menambahkan tugas ekspor ke aliran yang dikonfigurasi untuk menuliskan pesan status tentang tugas ekspor ke aliran status. Operasi append mengembalikan nomor urut yang mewakili ID tugas.
2. Komponen Greengrass membaca pesan secara berurutan dari aliran status, dan kemudian menyaring pesan berdasarkan nama aliran dan ID tugas atau berdasarkan properti tugas ekspor dari konteks pesan. Sebagai contoh, komponen Greengrass dapat memfilter dengan URL file input dari tugas ekspor, yang diwakili oleh objek `S3ExportTaskDefinition` dalam konteks pesan tersebut.

   Kode status berikut menunjukkan bahwa tugas ekspor telah mencapai keadaan lengkap:

   - `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 dihentikan karena definisi aliran atau ekspor dihapus, atau periode time-to-live (TTL) tugas berakhir.

> **Note**
>
> Tugas mungkin juga memiliki status `InProgress` atau `Warning`. Stream manager mengeluarkan peringatan ketika peristiwa mengembalikan kesalahan yang tidak mempengaruhi pelaksanaan tugas. Sebagai contoh, kegagalan untuk membersihkan sebagian unggahan akan mengembalikan peringatan.

3. Setelah tugas ekspor selesai, komponen Greengrass dapat menghapus file input yang sesuai.

Contoh berikut menunjukkan bagaimana komponen Greengrass mungkin membaca dan memproses pesan status.

Python

```
import time
from stream_manager import (
    ReadMessagesOptions,
    Status,
    StatusConfig,
    StatusLevel,
    StatusMessage,
    StreamManagerClient,
)
from stream_manager.util import Util

client = StreamManagerClient()

try:
    # Read the statuses from the export status stream
    is_file_uploaded_to_s3 = False
    while not is_file_uploaded_to_s3:
        try:
            messages_list = client.read_messages(
                "StatusStreamName", ReadMessagesOptions(min_message_count=1,
 read_timeout_millis=1000)
            )
            for message in messages_list:
                # Deserialize the status message first.
                status_message = Util.deserialize_json_bytes_to_obj(message.payload,
 StatusMessage)
```

```
                # Check the status of the status message. If the status is
 "Success",
                # the file was successfully uploaded to S3.
                # If the status was either "Failure" or "Cancelled", the server was
 unable to upload the file to S3.
                # We will print the message for why the upload to S3 failed from the
 status message.
                # If the status was "InProgress", the status indicates that the
 server has started uploading
                # the S3 task.
                if status_message.status == Status.Success:
                    logger.info("Successfully uploaded file at path " + file_url + "
 to S3.")
                    is_file_uploaded_to_s3 = True
                elif status_message.status == Status.Failure or
 status_message.status == Status.Canceled:
                    logger.info(
                        "Unable to upload file at path " + file_url + " to S3.
 Message: " + status_message.message
                    )
                    is_file_uploaded_to_s3 = True
            time.sleep(5)
        except StreamManagerException:
            logger.exception("Exception while running")
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

[Referensi Python SDK: read_messages | StatusMessage]

Java

```
import com.amazonaws.greengrass.streammanager.client.StreamManagerClient;
import com.amazonaws.greengrass.streammanager.client.StreamManagerClientFactory;
import com.amazonaws.greengrass.streammanager.client.utils.ValidateAndSerialize;
import com.amazonaws.greengrass.streammanager.model.ReadMessagesOptions;
import com.amazonaws.greengrass.streammanager.model.Status;
import com.amazonaws.greengrass.streammanager.model.StatusConfig;
import com.amazonaws.greengrass.streammanager.model.StatusLevel;
import com.amazonaws.greengrass.streammanager.model.StatusMessage;
```

```
try (final StreamManagerClient client =
StreamManagerClientFactory.standard().build()) {
   try {
       boolean isS3UploadComplete = false;
       while (!isS3UploadComplete) {
           try {
               // Read the statuses from the export status stream
               List<Message> messages = client.readMessages("StatusStreamName",
                   new
ReadMessagesOptions().withMinMessageCount(1L).withReadTimeoutMillis(1000L));
               for (Message message : messages) {
                   // Deserialize the status message first.
                   StatusMessage statusMessage =
ValidateAndSerialize.deserializeJsonBytesToObj(message.getPayload(),
StatusMessage.class);
                   // Check the status of the status message. If the status is
"Success", the file was successfully uploaded to S3.
                   // If the status was either "Failure" or "Canceled", the server
was unable to upload the file to S3.
                   // We will print the message for why the upload to S3 failed
from the status message.
                   // If the status was "InProgress", the status indicates that the
server has started uploading the S3 task.
                   if (Status.Success.equals(statusMessage.getStatus())) {
                       System.out.println("Successfully uploaded file at path " +
FILE_URL + " to S3.");
                       isS3UploadComplete = true;
                    } else if (Status.Failure.equals(statusMessage.getStatus()) ||
Status.Canceled.equals(statusMessage.getStatus())) {
                       System.out.println(String.format("Unable to upload file at
path %s to S3. Message %s",

statusMessage.getStatusContext().getS3ExportTaskDefinition().getInputUrl(),
                           statusMessage.getMessage()));
                       sS3UploadComplete = true;
                   }
               }
           } catch (StreamManagerException ignored) {
           } finally {
               // Sleep for sometime for the S3 upload task to complete before
trying to read the status message.
               Thread.sleep(5000);
           }
       } catch (e) {
```

```
        // Properly handle errors.
    }
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Referensi SDK Java: ReadMessages | StatusMessage

Node.js

```
const {
    StreamManagerClient, ReadMessagesOptions,
    Status, StatusConfig, StatusLevel, StatusMessage,
    util,
} = require('aws-greengrass-stream-manager-sdk');

const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        let isS3UploadComplete = false;
        while (!isS3UploadComplete) {
            try {
                // Read the statuses from the export status stream
                const messages = await c.readMessages("StatusStreamName",
                    new ReadMessagesOptions()
                        .withMinMessageCount(1)
                        .withReadTimeoutMillis(1000));

                messages.forEach((message) => {
                    // Deserialize the status message first.
                    const statusMessage =
 util.deserializeJsonBytesToObj(message.payload, StatusMessage);
                    // Check the status of the status message. If the status is
 'Success', the file was successfully uploaded to S3.
                    // If the status was either 'Failure' or 'Cancelled', the server
 was unable to upload the file to S3.
                    // We will print the message for why the upload to S3 failed
 from the status message.
                    // If the status was "InProgress", the status indicates that the
 server has started uploading the S3 task.
                    if (statusMessage.status === Status.Success) {
                        console.log(`Successfully uploaded file at path ${FILE_URL}
 to S3.`);
                        isS3UploadComplete = true;
```

```
                    } else if (statusMessage.status === Status.Failure ||
 statusMessage.status === Status.Canceled) {
                        console.log(`Unable to upload file at path ${FILE_URL} to
 S3. Message: ${statusMessage.message}`);
                        isS3UploadComplete = true;
                    }
                });
                // Sleep for sometime for the S3 upload task to complete before
 trying to read the status message.
                await new Promise((r) => setTimeout(r, 5000));
            } catch (e) {
                // Ignored
            }
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Referensi Node.js SDK: ReadMessages | StatusMessage

Konfigurasikan manajer pengaliran AWS IoT Greengrass

Pada perangkat inti Greengrass, stream manager dapat menyimpan, memproses, dan mengekspor data perangkat IoT. Strram manager menyediakan parameter yang Anda gunakan untuk mengonfigurasi pengaturan waktu aktif. Pengaturan ini berlaku untuk semua aliran pada perangkat inti Greengrass. Anda dapat menggunakan konsol AWS IoT Greengrass atau API untuk mengonfigurasi pengaturan stream manager ketika Anda men-deploy komponen. Perubahan berlaku setelah deployment selesai.

Parameter stream manager

Stream manager menyediakan parameter berikut yang dapat Anda konfigurasikan saat Anda men-deploy komponen tersebut ke perangkat inti Anda. Semua parameter bersifat opsional.

Direktori penyimpanan

Nama parameter: `STREAM_MANAGER_STORE_ROOT_DIR`

Jalur absolut dari folder lokal yang digunakan untuk menyimpan aliran. Nilai ini harus dimulai dengan garis miring ke depan (misalnya, `/data`).

Anda harus menentukan folder yang ada, dan pengguna sistem yang menjalankan komponen manajer aliran harus memiliki izin untuk membaca dan menulis ke folder ini. Misalnya, Anda dapat menjalankan perintah berikut untuk membuat dan mengkonfigurasi folder`/var/greengrass/streams`, yang Anda tentukan sebagai folder root stream manager. Perintah-perintah ini memungkinkan pengguna sistem default`ggc_user`,, untuk membaca dan menulis ke folder ini.

```
sudo mkdir /var/greengrass/streams
sudo chown ggc_user /var/greengrass/streams
sudo chmod 700 /var/greengrass/streams
```

Untuk informasi tentang cara mengamankan data aliran, lihat the section called “Keamanan data lokal”.

Default: *`/greengrass/v2`*`/work/aws.greengrass.StreamManager`

Port server

Nama parameter: `STREAM_MANAGER_SERVER_PORT`

Nomor port lokal yang digunakan untuk berkomunikasi dengan stream manager. Default-nya adalah `8088`.

Anda dapat menentukan `0` untuk menggunakan port yang tersedia acak.

Otentikasi klien

Nama parameter: `STREAM_MANAGER_AUTHENTICATE_CLIENT`

Menunjukkan apakah klien harus diautentikasi untuk berinteraksi dengan stream manager. Semua interaksi antara klien dan stream manager dikendalikan oleh Stream Manager SDK. Parameter ini menentukan klien mana yang dapat memanggil Manajer Pengaliran 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`. Memungkinkan hanya komponen Greengrass sebagai klien. Komponen menggunakan protokol internal inti AWS IoT Greengrass untuk diautentikasi dengan Stream Manager SDK.
- `false`. Memungkinkan setiap proses yang berjalan pada inti AWS IoT Greengrass untuk menjadi klien. Jangan tetapkan nilai ke `false` kecuali kasus bisnis Anda memerlukannya.

Misalnya, gunakan `false` hanya jika proses non-komponen pada perangkat inti harus berkomunikasi langsung dengan stream manager.

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.

Ukuran kolam benang

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 aliran mengonsumsi sumber daya perangkat keras, jadi cobalah untuk membatasi jumlah aliran ekspor pada perangkat yang dibatasi.

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 tumpukan default jika Anda berencana mengekspor sejumlah besar pengaliran.

Tingkat pencatatan

Nama parameter: `LOG_LEVEL`

Level logging untuk komponen. Pilih dari tingkat log berikut, yang tercantum di sini dalam urutan tingkat:

- `TRACE`
- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Default: `INFO`

Ukuran minimum untuk upload 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**
>
> Stream manager menggunakan properti `sizeThresholdForMultipartUploadBytes` aliran tersebut untuk menentukan apakah akan mengekspor ke Amazon S3 sebagai unggahan tunggal atau multipart. Komponen Greengrass yang ditentukan pengguna mengatur ambang batas ini ketika membuat aliran yang mengekspor ke Amazon S3. Ambang batas default adalah 5 MB.

Lihat juga

- Kelola aliran data di perangkat inti Greengrass
- Gunakan StreamManagerClient untuk bekerja dengan aliran
- Ekspor konfigurasi untuk tujuan AWS Cloud yang didukung

Lakukan inferensi machine learning

Dengan AWS IoT Greengrass, Anda dapat melakukan inferensi machine learning (ML) di perangkat edge Anda pada data yang dihasilkan secara lokal dengan menggunakan model yang terlatih cloud. Anda memperoleh manfaat dari latensi rendah dan penghematan biaya dari menjalankan inferensi lokal, tetapi masih mendapat manfaat dari daya komputasi cloud untuk model pelatihan dan pemrosesan yang kompleks.

AWS IoT Greengrassmembuat langkah-langkah yang diperlukan untuk melakukan inferensi lebih efisien. Anda dapat melatih model inferensi Anda di mana saja dan men-deploy-nya secara lokal sebagai komponen machine learning. Misalnya, Anda dapat membuat dan melatih model pembelajaran mendalam di Amazon SageMaker atau model visi komputer di Amazon Lookout for Vision. Kemudian, Anda dapat menyimpan model ini di bucket Amazon S3, sehingga Anda dapat menggunakan model ini sebagai artefak di komponen Anda untuk melakukan inferensi pada perangkat inti Anda.

Topik

Bagaimana inferensi ML AWS IoT Greengrass bekerja

AWSmenyediakan komponen pembelajaran mesin yang dapat Anda gunakan untuk membuat penerapan satu langkah untuk melakukan inferensi pembelajaran mesin di perangkat Anda. Anda juga dapat menggunakan komponen ini sebagai templat untuk membuat komponen kustom untuk memenuhi kebutuhan spesifik Anda.

AWSmenyediakan kategori komponen pembelajaran mesin berikut:

- Komponen model—Berisi model machine learning sebagai artefak Greengrass.
- Komponen waktu aktif—Berisi skrip yang menginstal kerangka kerja machine learning dan dependensinya pada perangkat inti Greengrass.
- Komponen Inferensi—Berisi kode inferensi dan mencakup dependensi komponen untuk menginstal kerangka machine learning dan mengunduh model machine learning yang telah dilatih sebelumnya.

Setiap penerapan yang Anda buat untuk melakukan inferensi pembelajaran mesin terdiri dari setidaknya satu komponen yang menjalankan aplikasi inferensi Anda, menginstal kerangka kerja pembelajaran mesin, dan mengunduh model pembelajaran mesin Anda. Untuk melakukan inferensi sampel dengan komponen AWS -provided, Anda menerapkan komponen inferensi ke perangkat inti Anda, yang secara otomatis menyertakan model dan komponen runtime yang sesuai sebagai dependensi. Untuk menyesuaikan penerapan, Anda dapat menyambungkan atau menukar komponen model sampel dengan komponen model khusus, atau Anda dapat menggunakan resep komponen untuk komponen yang AWS disediakan sebagai templat untuk membuat komponen inferensi, model, dan runtime kustom Anda sendiri.

Untuk melakukan inferensi pembelajaran mesin dengan menggunakan komponen khusus:

1. Buat komponen model. Komponen ini berisi model pembelajaran mesin yang ingin Anda gunakan untuk melakukan inferensi. AWSmenyediakan contoh model DLR dan Lite yang telah dilatih sebelumnya. TensorFlow Untuk menggunakan model kustom, buat komponen model Anda sendiri.
2. Buat komponen waktu aktif. Komponen ini berisi skrip yang diperlukan untuk menginstal runtime pembelajaran mesin untuk model Anda. AWSmenyediakan komponen runtime sampel untuk Deep Learning Runtime (DLR) dan Lite. TensorFlow Untuk menggunakan waktu aktif lain dengan model kustom Anda dan kode inferensi, buat komponen waktu aktif Anda sendiri.
3. Buat komponen inferensi. Komponen ini berisi kode inferensi Anda, dan menyertakan komponen model dan runtime Anda sebagai dependensi. AWSmenyediakan komponen inferensi sampel untuk klasifikasi gambar dan deteksi objek menggunakan DLR dan Lite. TensorFlow Untuk melakukan jenis inferensi lainnya, atau untuk menggunakan model dan waktu aktif kustom, buat komponen inferensi Anda sendiri.
4. Deploy komponen inferensi. Saat Anda menggunakan komponen ini, AWS IoT Greengrass juga secara otomatis men-deploy dependensi komponen model dan waktu aktif.

Untuk memulai komponen yang disediakan oleh AWS, lihat the section called “Lakukan contoh inferensi klasifikasi gambar”.

Untuk informasi selengkapnya tentang cara membuat komponen machine learning kustom, lihat Sesuaikan komponen machine learning Anda.

Apa yang berbeda di AWS IoT Greengrass versi 2?

AWS IoT Greengrassmengkonsolidasikan unit fungsional untuk pembelajaran mesin—seperti model, runtime, dan kode inferensi—ke dalam komponen yang memungkinkan Anda menggunakan proses satu langkah untuk menginstal runtime machine learning, mengunduh model terlatih, dan melakukan inferensi di perangkat Anda.

Dengan menggunakan komponen pembelajaran mesin AWS yang disediakan, Anda memiliki fleksibilitas untuk mulai melakukan inferensi pembelajaran mesin dengan kode inferensi sampel dan model yang telah dilatih sebelumnya. Anda dapat memasang komponen model kustom untuk menggunakan model terlatih kustom Anda sendiri dengan komponen inferensi dan waktu aktif yang disediakan oleh AWS. Untuk solusi machine learning yang sepenuhnya disesuaikan, Anda dapat menggunakan komponen publik sebagai templat untuk membuat komponen khusus dan menggunakan jenis waktu aktif, model, atau inferensi apa pun yang Anda inginkan.

Persyaratan

Untuk membuat dan menggunakan komponen pembelajaran mesin, Anda harus memiliki yang berikut:

- Sebuah perangkat inti Greengrass. Jika Anda tidak memilikinya, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2.
- Ruang penyimpanan lokal minimal 500 MB untuk menggunakan komponen machine learning sampel yang disediakan oleh AWS.

Sumber model yang didukung

AWS IoT Greengrass mendukung dengan menggunakan model machine learning yang terlatih khusus yang disimpan di Amazon S3. Anda juga dapat menggunakan pekerjaan pengemasan SageMaker tepi Amazon untuk langsung membuat komponen model untuk model yang SageMaker

dikompilasi NEO Anda. Untuk informasi tentang menggunakan SageMaker Edge Manager denganAWS IoT Greengrass, lihatGunakan Amazon SageMaker Edge Manager di perangkat inti Greengrass. Anda juga dapat menggunakan pekerjaan pengemasan model Lookout for Vision Amazon Lookout for Vision untuk membuat komponen model untuk model Lookout for Vision Anda. Untuk informasi selengkapnya tentang menggunakan Lookout for Vision AWS IoT Greengrass with, lihat. Gunakan Amazon Lookout for Vision

Bucket S3 yang berisi model Anda harus memenuhi persyaratan berikut:

- Mereka tidak boleh dienkripsi menggunakan SSE-C. Untuk bucket yang menggunakan enkripsi sisi server, inferensi pembelajaran AWS IoT Greengrass mesin saat ini hanya mendukung opsi enkripsi SSE-S3 atau SSE-KMS. Untuk informasi selengkapnya tentang opsi enkripsi sisi server, lihat Melindungi data menggunakan enkripsi sisi server di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
- Nama mereka tidak boleh menyertakan periode (`.`). Untuk informasi selengkapnya, lihat aturan tentang penggunaan bucket gaya hosted-virtual dengan SSL di Aturan untuk penamaan bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
- Bucket S3 yang menyimpan sumber model Anda harus berada di Akun AWS dan Wilayah AWS sebagai komponen machine learning.
- AWS IoT Greengrass harus memiliki izin `read` ke sumber model. Untuk memungkinkan AWS IoT Greengrass untuk mengakses bucket S3, peran perangkat Greengrass harus mengizinkan tindakan `s3:GetObject`. Untuk informasi lebih lanjut tentang peran perangkat, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Waktu aktif machine learning yang didukung

AWS IoT Greengrass memungkinkan Anda membuat komponen khusus untuk menggunakan waktu aktif machine learning pilihan Anda untuk melakukan inferensi machine learning dengan model yang terlatih khusus Anda. Untuk informasi selengkapnya tentang cara membuat komponen machine learning kustom, lihat Sesuaikan komponen machine learning Anda.

Untuk membuat proses memulai pembelajaran mesin lebih efisien, AWS IoT Greengrass berikan contoh inferensi, model, dan komponen runtime yang menggunakan runtime machine learning berikut:

- Deep Learning Runtime (DLR) v1.6.0 dan v1.3.0
- TensorFlow Lite v2.5.0

AWS-menyediakan komponen pembelajaran mesin

Tabel berikut mencantumkan komponen AWS yang disediakan yang digunakan untuk pembelajaran mesin.

> **Note**
>
> Beberapa komponen AWS yang disediakan bergantung pada versi minor spesifik dari inti Greengrass. Karena ketergantungan ini, Anda perlu memperbarui komponen ini saat memperbarui inti Greengrass ke versi minor baru. Untuk informasi tentang versi spesifik dari inti yang masing-masing komponen bergantung padanya, lihat topik komponen yang sesuai. Untuk informasi selengkapnya terkait cara memperbarui inti, lihat Perbarui perangkat lunak inti AWS IoT Greengrass (OTA).

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Lookout for Vision Edge Agent	Menyebarkan runtime Amazon Lookout for Vision pada perangkat inti Greengrass, sehingga Anda dapat menggunakan visi komputer untuk menemukan cacat pada produk industri.	Generik	Linux	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
SageMaker Manajer Tepi	Menyebarkan agen Amazon SageMaker Edge Manager di perangkat inti Greengrass.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Klasifikasi citra DLR	Komponen inferensi yang menggunakan penyimpanan model klasifikasi gambar DLR dan komponen waktu aktif DLR sebagai dependensi akan menginstal DLR, men-download model klasifikasi gambar sampel, dan melakukan inferensi klasifikasi gambar pada perangkat yang didukung.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Deteksi objek DLR	Komponen inferensi yang menggunakan penyimpanan model deteksi gambar DLR dan komponen waktu aktif DLR sebagai dependens i akan menginstal DLR, men-download sampel model deteksi, dan melakukan inferensi deteksi gambar pada perangkat yang didukung.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
Penyimpanan model klasifikasi gambar DLR	Komponen model yang berisi sampel ResNet -50 model klasifikasi gambar sebagai artefak Greengrass.	Generik	Linux, Windows	Tidak
Penyimpanan model deteksi DLR	Komponen model yang berisi sampel model deteksi objek YOLOv3 sebagai artefak Greengrass.	Generik	Linux, Windows	Tidak
Runtime DLR	Komponen waktu aktif yang berisi skrip instalasi yang digunakan untuk menginstal DLR dan dependensinya pada perangkat inti Greengrass.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
TensorFlow Klasifikasi gambar ringan	Komponen inferensi yang menggunakan penyimpanan model klasifikasi gambar TensorFlow Lite dan komponen runtime TensorFlow Lite sebagai dependensi untuk menginstal TensorFlow Lite, mengunduh model klasifikasi gambar sampel, dan melakukan inferensi klasifikasi gambar pada perangkat yang didukung.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
TensorFlow Deteksi objek Lite	Komponen inferensi yang menggunakan penyimpanan model deteksi objek TensorFlow Lite dan komponen runtime TensorFlow Lite sebagai dependensi untuk menginstal TensorFlow Lite, mengunduh model deteksi objek sampel, dan melakukan inferensi deteksi objek pada perangkat yang didukung.	Generik	Linux, Windows	Tidak

Komponen	Deskripsi	Jenis komponen	Sistem operasi yang didukung	Sumber terbuka
TensorFlow Toko model klasifikasi gambar Lite	Komponen model yang berisi contoh model MobileNet v1 sebagai artefak Greengrass.	Generik	Linux, Windows	Tidak
TensorFlow Toko model deteksi objek Lite	Komponen model yang berisi contoh model Single Shot Detection (SSD) sebagai MobileNet artefak Greengrass.	Generik	Linux, Windows	Tidak
TensorFlow Runtime ringan	Komponen runtime yang berisi skrip instalasi yang digunakan untuk menginstal TensorFlow Lite dan dependensinya pada perangkat inti Greengrass.	Generik	Linux, Windows	Tidak

Gunakan Amazon SageMaker Edge Manager di perangkat inti Greengrass

> **Important**
>
> SageMaker Edge Manager dihentikan pada 26 April 2024. Untuk informasi selengkapnya tentang melanjutkan penerapan model Anda ke perangkat edge, lihat SageMaker Edge Manager akhir masa pakai.

Amazon SageMaker Edge Manager adalah agen perangkat lunak yang berjalan pada perangkat edge. SageMaker Edge Manager menyediakan manajemen model untuk perangkat edge sehingga Anda dapat mengemas dan menggunakan model Amazon SageMaker Neo yang dikompilasi langsung di perangkat inti Greengrass. Dengan menggunakan SageMaker Edge Manager, Anda juga dapat mengambil sampel data input dan output model dari perangkat inti Anda, dan mengirim data tersebut ke AWS Cloud untuk pemantauan dan analisis. Karena SageMaker Edge Manager menggunakan SageMaker Neo untuk mengoptimalkan model Anda untuk perangkat keras target Anda, Anda tidak perlu menginstal runtime DLR langsung di perangkat Anda. Di perangkat Greengrass SageMaker , Edge Manager tidak memuat sertifikat AWS IoT lokal atau menghubungi AWS IoT titik akhir penyedia kredensi secara langsung. Sebagai gantinya, SageMaker Edge Manager menggunakan layanan pertukaran token untuk mengambil kredensi sementara dari titik akhir TES.

Bagian ini menjelaskan cara kerja SageMaker Edge Manager pada perangkat inti Greengrass.

Cara kerja SageMaker Edge Manager di perangkat Greengrass

Untuk menerapkan agen SageMaker Edge Manager ke perangkat inti Anda, buat penerapan yang menyertakan komponen. `aws.greengrass.SageMakerEdgeManager` AWS IoT Greengrassmengelola instalasi dan siklus hidup agen Edge Manager di perangkat Anda. Ketika versi baru dari agen biner tersedia, deploy versi terbaru dari komponen `aws.greengrass.SageMakerEdgeManager` untuk meningkatkan versi agen yang diinstal pada perangkat Anda.

Saat Anda menggunakan SageMaker Edge Manager denganAWS IoT Greengrass, alur kerja Anda mencakup langkah-langkah tingkat tinggi berikut:

1. Kompilasi model dengan SageMaker Neo.

2. Package model yang SageMaker dikompilasi NEO Anda menggunakan pekerjaan pengemasan SageMaker tepi. Ketika Anda menjalankan tugas pengemasan edge untuk model Anda, Anda dapat memilih untuk membuat komponen model dengan model kemasan sebagai artefak yang dapat digunakan untuk perangkat inti Greengrass Anda.
3. Buat komponen inferensi kustom. Anda menggunakan komponen inferensi ini untuk berinteraksi dengan agen Edge Manager untuk melakukan inferensi pada perangkat inti. Operasi ini meliputi pemuatan model, pemanggilan permintaan prediksi untuk menjalankan inferensi, dan pembongkaran model ketika komponen dimatikan.
4. Terapkan komponen SageMaker Edge Manager, komponen model yang dikemas, dan komponen inferensi untuk menjalankan model Anda di mesin SageMaker inferensi (agen Edge Manager) di perangkat Anda.

Untuk informasi selengkapnya tentang membuat pekerjaan kemasan tepi dan komponen inferensi yang berfungsi dengan SageMaker Edge Manager, lihat Menyebarkan Paket Model dan Agen Manajer Edge dengan AWS IoT Greengrass Panduan SageMaker Pengembang Amazon.

Tutorial: Memulai dengan SageMaker Edge ManagerTutorial menunjukkan cara mengatur dan menggunakan agen SageMaker Edge Manager pada perangkat inti Greengrass yang ada, AWS menggunakan kode contoh -provided yang dapat Anda gunakan untuk membuat inferensi sampel dan komponen model.

Saat Anda menggunakan SageMaker Edge Manager pada perangkat inti Greengrass, Anda juga dapat menggunakan fitur capture data untuk mengunggah data sampel ke file. AWS Cloud Capture data adalah SageMaker fitur yang Anda gunakan untuk mengunggah input inferensi, hasil inferensi, dan data inferensi tambahan ke bucket S3 atau direktori lokal untuk analisis masa depan. Untuk informasi selengkapnya tentang penggunaan data pengambilan dengan SageMaker Edge Manager, lihat Mengelola Model di Panduan SageMaker Pengembang Amazon.

Persyaratan

Anda harus memenuhi persyaratan berikut untuk menggunakan agen SageMaker Edge Manager pada perangkat inti Greengrass.

- Perangkat inti Greengrass yang berjalan di Amazon Linux 2, platform Linux berbasis Debian (x86_64 atau Armv8), atau Windows (x86_64). Jika Anda tidak memilikinya, lihat Tutorial: Memulai dengan AWS IoT Greengrass V2.

- Python 3.6 atau yang lebih baru, termasuk `pip` untuk versi Python Anda, diinstal pada perangkat inti anda.
- Peran perangkat Greengrass yang dikonfigurasi dengan berikut ini:
  - Hubungan kepercayaan yang memungkinkan `credentials.iot.amazonaws.com` dan `sagemaker.amazonaws.com` untuk meneruskan peran, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

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

  - Kebijakan yang dikelola AmazonSageMakerEdgeDeviceFleetPolicyIAM.
  - Tindakan `s3:PutObject`, seperti yang ditunjukkan dalam contoh kebijakan IAM berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
```

```
    }
  ]
}
```

- Bucket Amazon S3 yang dibuat sama Akun AWS dan Wilayah AWS sebagai perangkat inti Greengrass Anda. SageMaker Edge Manager memerlukan bucket S3 untuk membuat armada perangkat edge, dan menyimpan data sampel dari inferensi yang sedang berjalan di perangkat Anda. Untuk informasi selengkapnya tentang pembuatan bucket S3, lihat Memulai Amazon S3.
- Armada perangkat SageMaker edge yang menggunakan alias AWS IoT peran yang sama dengan perangkat inti Greengrass Anda. Untuk informasi selengkapnya, lihat Buat armada perangkat edge.
- Perangkat inti Greengrass Anda terdaftar sebagai perangkat tepi di armada perangkat Edge Anda. SageMaker Nama perangkat edge harus cocok dengan nama objek AWS IoT untuk perangkat inti Anda. Untuk informasi selengkapnya, lihat Daftarkan perangkat inti Greengrass Anda.

Memulai dengan SageMaker Edge Manager

Anda dapat menyelesaikan tutorial untuk mulai menggunakan SageMaker Edge Manager. Tutorial menunjukkan kepada Anda bagaimana memulai menggunakan SageMaker Edge Manager dengan komponen sampel AWS yang disediakan pada perangkat inti yang ada. Komponen sampel ini menggunakan komponen SageMaker Edge Manager sebagai dependensi untuk menyebarkan agen Edge Manager, dan melakukan inferensi menggunakan model pra-terlatih yang dikompilasi menggunakan Neo. SageMaker Lihat informasi yang lebih lengkap di Tutorial: Memulai dengan SageMaker Edge Manager.

Gunakan Amazon Lookout for Vision

> **Note**
>
> AWS IoT GreengrassSaat ini tidak mendukung fitur ini pada perangkat inti Windows.

Amazon Lookout for Vision adalahLayanan AWS yang dapat Anda gunakan untuk menemukan cacat visual pada produk industri. Ini menggunakan Vision untuk mengidentifikasi komponen yang hilang dalam produk industri, kerusakan pada kendaraan atau struktur, penyimpangan dalam jalur produksi, kapasitor yang hilang pada papan sirkuit cetak, dan cacat dalam wafer silikon. Untuk informasi selengkapnya, lihat Apa itu Amazon Lookout for Vision? di Panduan Pengembang Amazon Lookout for Vision.

Anda dapat membuat aplikasi Greengrass yang menggunakan Lookout for Vision inference untuk menemukan cacat visual pada perangkat inti Greengrass. Setelah Anda menerapkan alur kerja Lookout for Vision ke perangkat inti Greengrass, Anda dapat melakukan visi komputer tanpa koneksi ke layanan Lookout for Vision diAWS Cloud. Untuk membuat aplikasi Greengrass yang menggunakan Lookout for Vision, Anda menyiapkan dan menyebarkan komponen Greengrass berikut:

- Lookout for Vision - Berisi Lookout for Vision. Anda dapat menggunakan Lookout for Vision console dan API untuk menghasilkan komponen model yang mengemas model machine learning yang telah terlatih sebelumnya. Komponen ini adalah komponen Greengrass pribadi di AndaAkun AWS. Untuk informasi selengkapnya, lihat Membuat model Lookout for Vision dan Mengemas model Lookout for Vision di Panduan Pengembang Amazon Lookout for Vision.
- Lookout for Vision Edge Agent component - Menyediakan server runtime Lookout for Vision lokal yang menggunakan visi komputer untuk mendeteksi anomali menggunakan model machine learning yang Anda berikan. Komponen ini adalah komponenAWS -provided. Untuk informasi selengkapnya, lihat komponen Lookout for Vision.
- Lookout untuk komponen aplikasi klien Vision - Berinteraksi dengan komponen Lookout for Vision Edge Agent untuk memproses gambar untuk anomali. Anda dapat mengembangkan komponen aplikasi klien khusus yang mengirim gambar dan aliran video ke Lookout for Vision Edge Agent lokal dan melaporkan anomali apa pun yang dideteksi oleh model pembelajaran mesin. Untuk informasi selengkapnya, lihat Menulis komponen aplikasi klien dan Lookout for Vision Edge Agent API referensi di Panduan Pengembang Amazon Lookout for Vision.

Untuk informasi selengkapnya tentang cara membuat, mengonfigurasi, dan menggunakan komponen ini, lihat Menggunakan model Lookout for Vision pada perangkat edge di Panduan Pengembang Amazon Lookout for Vision.

Sesuaikan komponen machine learning Anda

Di AWS IoT Greengrass, Anda dapat mengonfigurasi sampel komponen machine learning untuk menyesuaikan cara Anda melakukan inferensi machine learning di perangkat Anda dengan komponen inferensi, model, dan waktu aktif sebagai blok bangunan. AWS IoT Greengrass juga menyediakan Anda fleksibilitas untuk menggunakan komponen sampel sebagai templat dan membuat komponen kustom Anda sendiri sesuai kebutuhan. Anda dapat mencampur dan mencocokkan pendekatan modular ini untuk menyesuaikan komponen inferensi machine learning Anda dengan cara berikut:

Menggunakan komponen inferensi sampel

- Ubah konfigurasi komponen inferensi ketika Anda men-deploy-nya.
- Gunakan model kustom dengan komponen inferensi sampel dengan mengganti komponen penyimpanan model sampel dengan komponen model kustom. Model kustom Anda harus dilatih menggunakan waktu aktif yang sama seperti model sampel.

Menggunakan komponen inferensi kustom

- Gunakan kode inferensi kustom dengan model sampel dan waktu aktif dengan menambahkan komponen model publik dan komponen waktu aktif sebagai dependensi komponen inferensi kustom.
- Buat dan tambahkan komponen model kustom atau komponen waktu aktif sebagai dependensi komponen inferensi kustom. Anda harus menggunakan komponen kustom jika Anda ingin menggunakan kode inferensi kustom atau waktu aktif yang tidak disediakan komponen sampelnya oleh AWS IoT Greengrass.

Topik

Ubah konfigurasi komponen inferensi publik

Di konsol AWS IoT Greengrass tersebut, halaman komponen menampilkan konfigurasi default komponen tersebut. Misalnya, konfigurasi default komponen klasifikasi gambar TensorFlow Lite terlihat seperti berikut:

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.TensorFlowLiteImageClassification:mqttproxy:1": {
        "policyDescription": "Allows access to publish via topic ml/tflite/image-
classification.",
        "operations": [
          "aws.greengrass#PublishToIoTCore"
        ],
        "resources": [
```

```
          "ml/tflite/image-classification"
        ]
      }
    }
  },
  "PublishResultsOnTopic": "ml/tflite/image-classification",
  "ImageName": "cat.jpeg",
  "InferenceInterval": 3600,
  "ModelResourceKey": {
    "model": "TensorFlowLite-Mobilenet"
  }
}
```

Ketika Anda men-deploy komponen inferensi publik, Anda dapat mengubah konfigurasi default untuk menyesuaikan deployment Anda. Untuk informasi tentang parameter konfigurasi yang tersedia untuk setiap komponen inferensi publik, lihat topik komponen di AWS-menyediakan komponen pembelajaran mesin.

Bagian ini menjelaskan cara men-deploy komponen yang dimodifikasi dari konsol AWS IoT Greengrass tersebut. Untuk informasi tentang men-deploy komponen menggunakan konsol AWS CLI, lihat Buat deployment.

Untuk men-deploy komponen inferensi publik yang dimodifikasi (konsol)

1. Masuk ke konsol AWS IoT Greengrass tersebut.
2. Pada menu navigasi, pilih Komponen.
3. Pada halaman Komponen, pada tab Komponen publik, pilih komponen yang ingin Anda deploy.
4. Pada halaman komponen, pilih Deploy.
5. Dari Tambahkan ke deployment, pilih salah satu langkah berikut ini:
   a. Untuk menggabungkan komponen ini ke deployment yang ada pada perangkat target Anda, pilih Tambahkan ke deployment yang ada, lalu pilih deployment yang ingin Anda revisi.
   b. Untuk membuat deployment baru di perangkat target Anda, pilih Buat deployment baru. Jika Anda memiliki deployment yang ada di perangkat, dengan memilih langkah ini Anda akan menggantikan deployment yang ada.
6. Di halaman Tentukan target, lakukan hal berikut:
   a. Di bawah informasi Deployment, masukkan atau ubah nama yang ramah untuk deployment Anda.

b. Di bawah Target deployment, pilih target untuk deployment Anda, dan pilih Selanjutnya. Anda tidak dapat mengubah target deployment jika Anda merevisi deployment yang ada.

7. Pada halaman Pilih komponen, di bawah Komponen publik verifikasi bahwa komponen inferensi dengan konfigurasi yang dimodifikasi dipilih, dan pilih Selanjutnya.
8. Pada halaman Konfigurasikan komponen, lakukan hal berikut:

   a. Pilih komponen inferensi, dan pilih Konfigurasikan komponen.

   b. Di bawah Pembaruan konfigurasi, masukkan nilai konfigurasi yang ingin Anda perbarui. Sebagai contoh, masukkan pembaruan konfigurasi berikut di kotak Konfigurasikan untuk menggabungkan untuk mengubah interval inferensi menjadi 15 detik, dan instruksikan komponen untuk mencari citra bernama `custom.jpg` di folder `/custom-ml-inference/images/`.

   ```
   {
     "InferenceInterval": "15",
     "ImageName": "custom.jpg",
     "ImageDirectory": "/custom-ml-inference/images/"
   }
   ```

   Untuk me-reset seluruh konfigurasi komponen ke nilai default-nya, tentukan string kosong tunggal `""` di kotak Reset path.

   c. Pilih Konfirmasi dan kemudian pilih Selanjutnya.
9. Pada halaman Konfigurasikan pengaturan lanjutan, simpan pengaturan konfigurasi default tersebut, dan pilih Selanjutnya.
10. Di halaman Tinjauan, pilih Deploy.

Gunakan model kustom dengan komponen inferensi sampel

Jika Anda ingin menggunakan komponen inferensi sampel dengan model machine learning Anda sendiri untuk waktu aktif yang untuknya AWS IoT Greengrass menyediakan komponen waktu aktif sampel, Anda harus mengganti komponen model publik dengan komponen yang menggunakan model-model tersebut sebagai artefak. Pada tingkat tinggi Anda menyelesaikan langkah-langkah berikut untuk menggunakan model kustom dengan komponen inferensi sampel:

1. Buat komponen model yang menggunakan model kustom dalam bucket S3 sebagai artefak. Model kustom Anda harus dilatih dengan menggunakan waktu aktif yang sama seperti model sampel.

2. Ubah parameter konfigurasi `ModelResourceKey` dalam komponen inferensi untuk menggunakan model kustom tersebut. Untuk informasi tentang pembaruan konfigurasi komponen inferensi, lihat Ubah konfigurasi komponen inferensi publik

Ketika Anda men-deploy komponen inferensi, AWS IoT Greengrass akan mencari versi terbaru dari dependensi komponennya. Ia akan menimpa komponen model publik dependen jika versi kustom selanjutnya dari komponen tersebut berada di Akun AWS dan Wilayah AWS yang sama.

Buat komponen model kustom (konsol)

1. Unggah model Anda ke bucket S3. Untuk informasi tentang mengunggah model ke bucket S3, lihat Bekerja dengan Bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

   > **Note**
   >
   > Anda harus menyimpan artefak Anda di bucket S3 yang berada di Akun AWS dan Wilayah AWS yang sama dengan komponen tersebut. Untuk memungkinkan AWS IoT Greengrass untuk mengakses artefak ini, peran perangkat Greengrass harus mengizinkan tindakan `s3:GetObject`. Untuk informasi lebih lanjut tentang peran perangkat, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

2. Pada menu navigasi konsol AWS IoT Greengrass tersebut, pilih Komponen.
3. Ambil resep komponen untuk komponen penyimpanan model publik.

   a. Pada halaman Komponen, pada tab Komponen publik, cari dan pilih komponen model publik yang ingin Anda buat versi barunya. Misalnya, `variant.DLR.ImageClassification.ModelStore`.

   b. Pada halaman komponen, pilih Lihat resep dan salin resep JSON yang ditampilkan.
4. Pada halaman Komponen, pada tab Komponen saya, pilih Buat komponen.
5. Pada halaman Buat Komponen, di bawah Informasi komponen, pilih Masukkan resep sebagai JSON sebagai sumber komponen Anda.
6. Di kotak Resep, tempelkan komponen resep yang sebelumnya telah Anda salin.
7. Dalam resep tersebut, perbarui nilai berikut:

   - `ComponentVersion`: Kenaikan versi minor komponen.

Ketika Anda membuat komponen kustom untuk menimpa komponen model publik, Anda harus memperbarui hanya versi minor dari versi komponen yang ada. Sebagai contoh, jika versi komponen publik `2.1.0` Anda dapat membuat komponen kustom dengan versi `2.1.1`.

- `Manifests.Artifacts.Uri`: Perbarui setiap nilai URI untuk Amazon S3 URI model yang ingin Anda gunakan.

> **Note**
>
> Jangan ubah nama komponen.

8. Pilih Buat komponen.

Buat komponen model kustom (AWS CLI)

1. Unggah model Anda ke bucket S3. Untuk informasi tentang mengunggah model ke bucket S3, lihat Bekerja dengan Bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Anda harus menyimpan artefak Anda di bucket S3 yang berada di Akun AWS dan Wilayah AWS yang sama dengan komponen tersebut. Untuk memungkinkan AWS IoT Greengrass untuk mengakses artefak ini, peran perangkat Greengrass harus mengizinkan tindakan `s3:GetObject`. Untuk informasi lebih lanjut tentang peran perangkat, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

2. Jalankan perintah berikut untuk mengambil resep komponen pada komponen publik. Perintah ini menuliskan resep komponen untuk file output yang Anda berikan dalam perintah Anda. Konversi string berkode base64 yang diambil untuk JSON atau YAML, sesuai keperluan.

   Linux, macOS, or Unix

```
aws greengrassv2 get-component \
    --arn <arn> \
    --recipe-output-format <recipe-format> \
    --query recipe \
    --output text | base64 --decode > <recipe-file>
```

Windows Command Prompt (CMD)

```
aws greengrassv2 get-component ^
    --arn <arn> ^
    --recipe-output-format <recipe-format> ^
    --query recipe ^
    --output text > <recipe-file>.base64

certutil -decode <recipe-file>.base64 <recipe-file>
```

PowerShell

```
aws greengrassv2 get-component `
    --arn <arn> `
    --recipe-output-format <recipe-format> `
    --query recipe `
    --output text > <recipe-file>.base64

certutil -decode <recipe-file>.base64 <recipe-file>
```

3. Perbarui nama file resep untuk *<component-name>*-*<component-version>*, di mana versi komponen adalah versi target komponen baru. Misalnya, `variant.DLR.ImageClassification.ModelStore-2.1.1.yaml`.
4. Dalam resep tersebut, perbarui nilai berikut:

   - `ComponentVersion`: Kenaikan versi minor komponen.

     Ketika Anda membuat komponen kustom untuk menimpa komponen model publik, Anda harus memperbarui hanya versi minor dari versi komponen yang ada. Sebagai contoh, jika versi komponen publik `2.1.0` Anda dapat membuat komponen kustom dengan versi `2.1.1`.
   - `Manifests.Artifacts.Uri`: Perbarui setiap nilai URI untuk Amazon S3 URI model yang ingin Anda gunakan.

   > **Note**
   >
   > Jangan ubah nama komponen.

5. Jalankan perintah berikut untuk membuat komponen baru dengan menggunakan resep yang telah Anda ambil dan ubah.

```
aws greengrassv2 create-component-version \
    --inline-recipe fileb://path/to/component/recipe
```

> **Note**
>
> Langkah ini membuat komponen dalam layanan AWS IoT Greengrass dalam aplikasi AWS Cloud. Anda dapat menggunakan Greengrass CLI untuk mengembangkan, menguji, dan men-deploy komponen Anda secara lokal sebelum Anda meng-upload ke cloud. Untuk informasi selengkapnya, lihat Kembangkan AWS IoT Greengrass komponen.

Untuk informasi selengkapnya tentang membuat komponen, lihat Kembangkan AWS IoT Greengrass komponen.

Buat komponen machine learning khusus

Anda harus membuat komponen kustom jika Anda ingin menggunakan kode inferensi kustom atau waktu aktif yang tidak disediakan komponen sampelnya oleh AWS IoT Greengrass. Anda dapat menggunakan kode inferensi kustom Anda dengan sampel model machine learning dan waktu aktif yang disediakan oleh AWS, atau Anda dapat mengembangkan solusi inferensi machine learning yang benar-benar disesuaikan dengan model dan waktu aktif Anda sendiri. Jika model Anda menggunakan waktu aktif yang untuknya AWS IoT Greengrass menyediakan komponen waktu aktif sampel, Anda dapat menggunakan komponen waktu aktif, dan Anda perlu membuat komponen kustom hanya untuk kode inferensi Anda dan model yang ingin Anda gunakan.

Topik

- Ambil resep untuk komponen publik
- Ambil contoh artefak komponen
- Unggah artefak komponen ke bucket S3
- Buat komponen khusus

Ambil resep untuk komponen publik

Anda dapat menggunakan resep komponen machine learning publik yang ada sebagai templat untuk membuat komponen kustom. Untuk melihat resep komponen untuk versi terbaru dari komponen publik, gunakan konsol atau AWS CLI sebagai berikut:

- Menggunakan konsol

  1. Pada halaman Komponen, pada tab Komponen publik, cari dan pilih komponen model publik yang ingin Anda buat versi barunya.
  2. Pada halaman komponen, pilih Lihat resep.

- Menggunakan AWS CLI

  Jalankan perintah berikut untuk mengambil resep komponen pada komponen varian publik. Perintah ini menuliskan resep komponen untuk file resep JSON atau YAML yang Anda berikan dalam perintah Anda.

  Linux, macOS, or Unix

  ```
  aws greengrassv2 get-component \
      --arn <arn> \
      --recipe-output-format <recipe-format> \
      --query recipe \
      --output text | base64 --decode > <recipe-file>
  ```

  Windows Command Prompt (CMD)

  ```
  aws greengrassv2 get-component ^
      --arn <arn> ^
      --recipe-output-format <recipe-format> ^
      --query recipe ^
      --output text > <recipe-file>.base64

  certutil -decode <recipe-file>.base64 <recipe-file>
  ```

  PowerShell

  ```
  aws greengrassv2 get-component `
      --arn <arn> `
      --recipe-output-format <recipe-format> `
      --query recipe `
  ```

```
    --output text > <recipe-file>.base64

certutil -decode <recipe-file>.base64 <recipe-file>
```

Ganti nilai-nilai dalam perintah Anda sebagai berikut:

- *<arn>*. Amazon Resource Name (ARN) dari komponen publik.
- *<recipe-format>*. Format di mana Anda ingin membuat file resep. Nilai yang didukung adalah `JSON` dan `YAML`.
- *<recipe-file>*. Nama resep dalam format *<component-name>*-*<component-version>*.

Ambil contoh artefak komponen

Anda dapat menggunakan artefak yang digunakan oleh komponen machine learning publik sebagai templat untuk membuat artefak komponen kustom Anda, seperti kode inferensi atau skrip instalasi waktu aktif.

Untuk melihat artefak sampel yang disertakan dalam komponen machine learning publik, deploy komponen inferensi publik dan kemudian lihat artefak pada perangkat Anda di folder */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/`*component-version*`/`.

Unggah artefak komponen ke bucket S3

Sebelum Anda dapat membuat komponen kustom, Anda harus meng-upload artefak komponen ke bucket S3 dan menggunakan URI S3 dalam resep komponen Anda. Misalnya, untuk menggunakan kode inferensi kustom dalam komponen inferensi Anda, unggah kode tersebut ke bucket S3. Anda kemudian dapat menggunakan Amazon S3 URI kode inferensi Anda sebagai artefak dalam komponen Anda.

Untuk informasi tentang mengunggah konten ke bucket S3, lihat Bekerja dengan Bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Anda harus menyimpan artefak Anda di bucket S3 yang berada di Akun AWS dan Wilayah AWS yang sama dengan komponen tersebut. Untuk memungkinkan AWS IoT Greengrass untuk mengakses artefak ini, peran perangkat Greengrass harus mengizinkan tindakan `s3:GetObject`. Untuk informasi lebih lanjut tentang peran perangkat, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Buat komponen khusus

Anda dapat menggunakan artefak dan resep yang telah Anda ambil untuk membuat komponen machine learning kustom Anda. Lihat contohnya di Buat komponen inferensi khusus.

Untuk informasi detail tentang membuat dan men-deploy komponen ke perangkat Greengrass, lihat Kembangkan AWS IoT Greengrass komponen dan Deploy komponen AWS IoT Greengrass ke perangkat.

Buat komponen inferensi khusus

Bagian ini menunjukkan kepada Anda cara membuat komponen inferensi kustom dengan menggunakan komponen klasifikasi gambar DLR sebagai templat.

Topik

- Unggah kode inferensi Anda ke bucket Amazon S3
- Buat resep untuk komponen inferensi Anda
- Buat komponen inferensi

Unggah kode inferensi Anda ke bucket Amazon S3

Buat kode inferensi Anda dan kemudian unggah ke bucket S3. Untuk informasi tentang mengunggah konten ke bucket S3, lihat Bekerja dengan Bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

> **Note**
>
> Anda harus menyimpan artefak Anda di bucket S3 yang berada di Akun AWS dan Wilayah AWS yang sama dengan komponen tersebut. Untuk memungkinkan AWS IoT Greengrass untuk mengakses artefak ini, peran perangkat Greengrass harus mengizinkan tindakan `s3:GetObject`. Untuk informasi lebih lanjut tentang peran perangkat, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Buat resep untuk komponen inferensi Anda

1. Jalankan perintah berikut untuk mengambil resep komponen pada komponen klasifikasi citra DLR. Perintah ini menuliskan resep komponen untuk file resep JSON atau YAML yang Anda berikan dalam perintah Anda.

   Linux, macOS, or Unix

   ```
   aws greengrassv2 get-component \
       --arn
    arn:aws:greengrass:region:aws:components:aws.greengrass.DLRImageClassification:versions
    \
       --recipe-output-format JSON | YAML \
       --query recipe \
       --output text | base64 --decode > <recipe-file>
   ```

   Windows Command Prompt (CMD)

   ```
   aws greengrassv2 get-component ^
       --arn
    arn:aws:greengrass:region:aws:components:aws.greengrass.DLRImageClassification:versions
    ^
       --recipe-output-format JSON | YAML ^
       --query recipe ^
       --output text > <recipe-file>.base64

   certutil -decode <recipe-file>.base64 <recipe-file>
   ```

   PowerShell

   ```
   aws greengrassv2 get-component `
       --arn
    arn:aws:greengrass:region:aws:components:aws.greengrass.DLRImageClassification:versions
    `
       --recipe-output-format JSON | YAML `
       --query recipe `
       --output text > <recipe-file>.base64

   certutil -decode <recipe-file>.base64 <recipe-file>
   ```

Ganti *<recipe-file>* dengan nama resep dalam format *<component-name>*-*<component-version>*.

2. Di objek `ComponentDependencies` dalam resep Anda, lakukan salah satu atau lebih hal berikut ini tergantung pada model dan waktu aktif komponen yang ingin Anda gunakan:

   - Jaga dependensi komponen DLR jika Anda ingin menggunakan model terkompilasi DLR. Anda juga dapat menggantinya dengan dependensi pada komponen waktu aktif kustom, seperti yang ditunjukkan dalam contoh berikut.

     Komponen runtime

     JSON

     ```
     {
         "<runtime-component>": {
             "VersionRequirement": "<version>",
             "DependencyType": "HARD"
         }
     }
     ```

     YAML

     ```
     <runtime-component>:
         VersionRequirement: "<version>"
         DependencyType: HARD
     ```

   - Pertahankan dependensi penyimpanan model klasifikasi gambar DLR untuk menggunakan model ResNet -50 yang telah dilatih sebelumnya yang AWS menyediakan, atau memodifikasinya untuk menggunakan komponen model kustom. Ketika Anda menyertakan dependensi untuk komponen model publik, jika versi kustom komponen ada di Akun AWS dan Wilayah AWS, maka komponen inferensi menggunakan komponen kustom. Tentukan dependensi komponen model seperti yang ditunjukkan dalam contoh berikut.

     Komponen model publik

     JSON

     ```
     {
         "variant.DLR.ImageClassification.ModelStore": {
             "VersionRequirement": "<version>",
     ```

```
        "DependencyType": "HARD"
    }
}
```

YAML

```
variant.DLR.ImageClassification.ModelStore:
    VersionRequirement: "<version>"
    DependencyType: HARD
```

Komponen model kustom

JSON

```
{
    "<custom-model-component>": {
        "VersionRequirement": "<version>",
        "DependencyType": "HARD"
    }
}
```

YAML

```
<custom-model-component>:
    VersionRequirement: "<version>"
    DependencyType: HARD
```

3. Di objek `ComponentConfiguration`, tambahkan konfigurasi default untuk komponen ini. Anda nanti dapat mengubah konfigurasi ini ketika Anda men-deploy komponen tersebut. Kutipan berikut menunjukkan konfigurasi komponen untuk komponen klasifikasi gambar DLR.

   Misalnya, jika Anda menggunakan komponen model kustom sebagai dependensi untuk komponen inferensi kustom Anda, maka ubah `ModelResourceKey` untuk memberikan nama-nama model yang Anda gunakan.

   JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.ImageClassification:mqttproxy:1": {
```

```
        "policyDescription": "Allows access to publish via topic ml/dlr/image-
classification.",
        "operations": [
          "aws.greengrass#PublishToIoTCore"
        ],
        "resources": [
          "ml/dlr/image-classification"
        ]
      }
    }
  },
  "PublishResultsOnTopic": "ml/dlr/image-classification",
  "ImageName": "cat.jpeg",
  "InferenceInterval": 3600,
  "ModelResourceKey": {
    "armv7l": "DLR-resnet50-armv7l-cpu-ImageClassification",
    "x86_64": "DLR-resnet50-x86_64-cpu-ImageClassification",
    "aarch64": "DLR-resnet50-aarch64-cpu-ImageClassification"
  }
}
```

YAML

```
accessControl:
    aws.greengrass.ipc.mqttproxy:
        'aws.greengrass.ImageClassification:mqttproxy:1':
            policyDescription: 'Allows access to publish via topic ml/dlr/image-
classification.'
            operations:
                - 'aws.greengrass#PublishToIoTCore'
            resources:
                - ml/dlr/image-classification
PublishResultsOnTopic: ml/dlr/image-classification
ImageName: cat.jpeg
InferenceInterval: 3600
ModelResourceKey:
    armv7l: "DLR-resnet50-armv7l-cpu-ImageClassification"
    x86_64: "DLR-resnet50-x86_64-cpu-ImageClassification"
    aarch64: "DLR-resnet50-aarch64-cpu-ImageClassification"
```

4. Di objek `Manifests`, berikan informasi tentang artefak tersebut dan konfigurasi komponen ini yang digunakan ketika komponen tersebut di-deploy pada beberapa platform yang berbeda dan informasi lain yang diperlukan untuk berhasil menjalankan komponen tersebut. Kutipan berikut

menunjukkan konfigurasi objek `Manifests` untuk platform Linux dalam komponen klasifikasi gambar DLR.

JSON

```
{
  "Manifests": [
    {
      "Platform": {
        "os": "linux",
        "architecture": "arm"
      },
      "Name": "32-bit armv7l - Linux (raspberry pi)",
      "Artifacts": [
        {
          "URI": "s3://SAMPLE-BUCKET/sample-artifacts-directory/
image_classification.zip",
          "Unarchive": "ZIP"
        }
      ],
      "Lifecycle": {
        "Setenv": {
          "DLR_IC_MODEL_DIR":
 "{variant.DLR.ImageClassification.ModelStore:artifacts:decompressedPath}/
{configuration:/ModelResourceKey/armv7l}",
          "DEFAULT_DLR_IC_IMAGE_DIR": "{artifacts:decompressedPath}/
image_classification/sample_images/"
        },
        "run": {
          "RequiresPrivilege": true,
          "script": ". {variant.DLR:configuration:/MLRootPath}/
greengrass_ml_dlr_venv/bin/activate\npython3 {artifacts:decompressedPath}/
image_classification/inference.py"
        }
      }
    }
  ]
}
```

YAML

```
Manifests:
  - Platform:
```

```
      os: linux
      architecture: arm
    Name: 32-bit armv7l - Linux (raspberry pi)
    Artifacts:
      - URI: s3://SAMPLE-BUCKET/sample-artifacts-directory/
image_classification.zip
        Unarchive: ZIP
    Lifecycle:
      Setenv:
        DLR_IC_MODEL_DIR:
 "{variant.DLR.ImageClassification.ModelStore:artifacts:decompressedPath}/
{configuration:/ModelResourceKey/armv7l}"
        DEFAULT_DLR_IC_IMAGE_DIR: "{artifacts:decompressedPath}/
image_classification/sample_images/"
      run:
        RequiresPrivilege: true
        script: |-
          . {variant.DLR:configuration:/MLRootPath}/greengrass_ml_dlr_venv/bin/
activate
          python3 {artifacts:decompressedPath}/image_classification/inference.py
```

Untuk informasi detail tentang membuat resep komponen, lihat AWS IoT Greengrass referensi resep komponen.

Buat komponen inferensi

Gunakan konsol AWS IoT Greengrass atau konsol AWS CLI untuk membuat komponen dengan menggunakan resep yang baru saja Anda tetapkan. Setelah Anda membuat komponen, Anda dapat men-deploy-nya untuk melakukan inferensi pada perangkat Anda. Untuk contoh cara menggunakan komponen inferensi, lihat Tutorial: Lakukan inferensi klasifikasi gambar sampel menggunakan Lite TensorFlow .

Buat komponen inferensi kustom (konsol)

1. Masuk ke konsol AWS IoT Greengrass tersebut.
2. Pada menu navigasi, pilih Komponen.
3. Pada halaman Komponen, pada tab Komponen saya, pilih Buat komponen.
4. Pada halaman Buat Komponen, di bawah Informasi komponen, pilih Masukkan resep sebagai JSON atau Masukkan resep sebagai YAML sebagai sumber komponen Anda.
5. Di kotak Resep, masukkan resep kustom yang Anda buat.

6. Klik Buat Komponen.

Buat komponen inferensi khusus (AWS CLI)

Jalankan perintah berikut untuk membuat komponen baru dengan menggunakan resep yang telah Anda buat.

```
aws greengrassv2 create-component-version \
    --inline-recipe fileb://path/to/recipe/file
```

> **Note**
>
> Langkah ini membuat komponen dalam layanan AWS IoT Greengrass di AWS Cloud. Anda dapat menggunakan Greengrass CLI untuk mengembangkan, menguji, dan men-deploy komponen Anda secara lokal sebelum Anda meng-upload ke cloud. Lihat informasi yang lebih lengkap di Kembangkan AWS IoT Greengrass komponen.

Menyelesaikan masalah inferensi machine learning

Gunakan informasi pemecahan masalah dan solusi di bagian ini untuk membantu menyelesaikan masalah dengan komponen machine learning Anda. Untuk komponen inferensi pembelajaran mesin publik, lihat pesan kesalahan di log komponen berikut:

Linux or Unix

- */greengrass/v2*/logs/aws.greengrass.DLRImageClassification.log
- */greengrass/v2*/logs/aws.greengrass.DLRObjectDetection.log
- */greengrass/v2*/logs/aws.greengrass.TensorFlowLiteImageClassification.log
- */greengrass/v2*/logs/aws.greengrass.TensorFlowLiteObjectDetection.log

Windows

- *C:\greengrass\v2*\logs\aws.greengrass.DLRImageClassification.log
- *C:\greengrass\v2*\logs\aws.greengrass.DLRObjectDetection.log

- *C:\greengrass\v2*\logs\aws.greengrass.TensorFlowLiteImageClassification.log
- *C:\greengrass\v2*\logs\aws.greengrass.TensorFlowLiteObjectDetection.log

Jika komponen diinstal dengan benar, maka log komponen akan berisi lokasi pustaka yang digunakannya untuk inferensi.

Masalah

- Gagal mengambil pustaka
- Cannot open shared object file
- Error: ModuleNotFoundError: No module named '<library>'
- Tidak ada perangkat berkemampuan CUDA yang terdeteksi
- Tidak ada file atau direktori seperti itu
- RuntimeError: module compiled against API version 0xf but this version of NumPy is <version>
- picamera.exc.PiCameraError: Camera is not enabled
- Kesalahan memori
- Kesalahan ruang disk
- Batas waktu mengalami kesalahan

Gagal mengambil pustaka

Kesalahan berikut terjadi ketika skrip penginstal gagal mengunduh pustaka yang diperlukan selama penerapan pada perangkat Raspberry Pi.

```
Err:2 http://raspbian.raspberrypi.org/raspbian buster/main armhf python3.7-dev armhf
 3.7.3-2+deb10u1
404 Not Found [IP: 93.93.128.193 80]
E: Failed to fetch http://raspbian.raspberrypi.org/raspbian/pool/main/p/python3.7/
libpython3.7-dev_3.7.3-2+deb10u1_armhf.deb 404 Not Found [IP: 93.93.128.193 80]
```

Jalankan `sudo apt-get update` dan deploy komponen Anda lagi.

Cannot open shared object file

Anda mungkin melihat kesalahan yang mirip dengan berikut ini ketika skrip penginstal gagal mengunduh dependensi yang diperlukan untuk `opencv-python` selama deployment pada perangkat Raspberry Pi.

```
ImportError: libopenjp2.so.7: cannot open shared object file: No such file or directory
```

Jalankan perintah berikut untuk secara manual menginstal dependensi untuk `opencv-python`:

```
sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
 libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
```

Error: ModuleNotFoundError: No module named '<library>'

Anda mungkin melihat kesalahan ini di log komponen runtime HTML (`variant.DLR.log`atau`variant.TensorFlowLite.log`) saat pustaka runtime HTML atau dependensinya tidak diinstal dengan benar. Kesalahan ini dapat terjadi dalam kasus-kasus berikut:

- Jika Anda menggunakan `UseInstaller` opsi, yang diaktifkan secara default, kesalahan ini menunjukkan bahwa komponen runtime HTML gagal menginstal runtime atau dependensinya. Lakukan hal-hal berikut:
  1. Konfigurasikan komponen runtime ML untuk menonaktifkan `UseInstaller` opsi.
  2. Instal runtime ML dan dependensinya, dan buat mereka tersedia untuk pengguna sistem yang menjalankan komponen ML. Untuk informasi selengkapnya, lihat hal berikut:
     - Opsi runtime DLR UseInstaller
     - TensorFlowOpsi runtime UseInstaller Lite
- Jika Anda tidak menggunakan `UseInstaller` opsi, kesalahan ini menunjukkan bahwa runtime HTML atau dependensinya tidak diinstal untuk pengguna sistem yang menjalankan komponen ML. Lakukan hal-hal berikut:
  1. Periksa apakah pustaka diinstal untuk pengguna sistem yang menjalankan komponen ML. *`Ganti ggc_user`* dengan nama pengguna sistem, dan ganti *`tflite_runtime`* dengan nama perpustakaan untuk diperiksa.

     Linux or Unix

     ```
     sudo -H -u ggc_user bash -c "python3 -c 'import tflite_runtime'"
     ```

Windows

```
runas /user:ggc_user "py -3 -c \"import tflite_runtime\""
```

2. Jika pustaka tidak diinstal, instal untuk pengguna itu. *Ganti ggc_user* dengan nama pengguna sistem, dan ganti *tflite_runtime* dengan nama pustaka.

   Linux or Unix

   ```
   sudo -H -u ggc_user bash -c "python3 -m pip install --user tflite_runtime"
   ```

   Windows

   ```
   runas /user:ggc_user "py -3 -m pip install --user tflite_runtime"
   ```

   Untuk informasi selengkapnya tentang dependensi untuk setiap runtime ML, lihat berikut ini:

   - Opsi runtime DLR UseInstaller
   - TensorFlowOpsi runtime UseInstaller Lite

3. Jika masalah berlanjut, instal pustaka untuk pengguna lain untuk mengonfirmasi apakah perangkat ini dapat menginstal pustaka. Pengguna dapat berupa, misalnya, pengguna Anda, pengguna root, atau pengguna administrator. Jika Anda tidak berhasil menginstal pustaka untuk pengguna mana pun, perangkat Anda mungkin tidak mendukung pustaka. Konsultasikan dokumentasi perpustakaan untuk meninjau persyaratan dan memecahkan masalah penginstalan.

Tidak ada perangkat berkemampuan CUDA yang terdeteksi

Anda mungkin melihat kesalahan berikut saat menggunakan akselerasi GPU. Jalankan perintah berikut untuk mengaktifkan akses GPU untuk pengguna Greengrass.

```
sudo usermod -a -G video ggc_user
```

Tidak ada file atau direktori seperti itu

Kesalahan berikut menunjukkan bahwa komponen waktu aktif tidak dapat mengatur lingkungan virtual dengan benar:

- *MLRootPath*/greengrass_ml_dlr_conda/bin/conda: No such file or directory
- *MLRootPath*/greengrass_ml_dlr_venv/bin/activate: No such file or directory
- *MLRootPath*/greengrass_ml_tflite_conda/bin/conda: No such file or directory
- *MLRootPath*/greengrass_ml_tflite_venv/bin/activate: No such file or directory

Periksa log untuk memastikan bahwa semua waktu aktif dependensi diinstal dengan benar. Untuk informasi selengkapnya tentang pustaka yang diinstal oleh skrip penginstal, lihat topik berikut ini:

- Runtime DLR
- TensorFlow Runtime ringan

Secara default, *ML RootPath* diatur ke*/greengrass/v2*/work/*component-name*/greengrass_ml. Untuk mengubah lokasi ini, sertakan Runtime DLR atau komponen waktu aktif TensorFlow Runtime ringan secara langsung dalam deployment Anda, dan tentukan nilai yang diubah untuk parameter MLRootPath dalam pembaruan gabungan konfigurasi. Untuk informasi lebih lanjut tentang mengonfigurasi komponen, lihat Perbarui konfigurasi komponen.

> **Note**
>
> Untuk komponen DLR v1.3.x, Anda mengatur parameter MLRootPath dalam konfigurasi komponen inferensi, dan nilai defaultnya adalah $HOME/greengrass_ml.

RuntimeError: module compiled against API version 0xf but this version of NumPy is <version>

Anda mungkin melihat kesalahan berikut saat menjalankan inferensi pembelajaran mesin pada Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye.

```
RuntimeError: module compiled against API version 0xf but this version of numpy is 0xd
ImportError: numpy.core.multiarray failed to import
```

Kesalahan ini terjadi karena Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy dari versi yang dibutuhkan OpenCV. Untuk memperbaiki masalah ini, jalankan perintah berikut untuk meningkatkan NumPy ke versi terbaru.

```
pip3 install --upgrade numpy
```

picamera.exc.PiCameraError: Camera is not enabled

Anda mungkin melihat kesalahan berikut saat menjalankan inferensi pembelajaran mesin pada Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye.

```
picamera.exc.PiCameraError: Camera is not enabled. Try running 'sudo raspi-config' and
 ensure that the camera has been enabled.
```

Kesalahan ini terjadi karena Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang tidak kompatibel dengan komponen ML. Untuk memperbaiki masalah ini, aktifkan tumpukan kamera lama.

Untuk mengaktifkan tumpukan kamera lama

1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

   ```
   sudo raspi-config
   ```
2. Pilih Opsi Antarmuka.
3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.
4. Reboot Raspberry Pi.

Kesalahan memori

Kesalahan berikut biasanya terjadi ketika perangkat tidak memiliki cukup memori dan proses komponen terganggu.

- `stderr. Killed.`
- `exitCode=137`

Kami merekomendasikan minimal 500 MB memori untuk men-deploy komponen inferensi machine learning.

Kesalahan ruang disk

Kesalahan `no space left on device` biasanya terjadi ketika sebuah klaster tidak memiliki penyimpanan yang cukup. Pastikan bahwa ada cukup ruang disk yang tersedia di perangkat Anda sebelum Anda menyebarkan komponen lagi. Kami merekomendasikan minimal 500 MB memori untuk men-deploy komponen inferensi machine learning publik.

Batas waktu mengalami kesalahan

Komponen machine learning publik mengunduh file model machine learning besar yang lebih besar dari 200 MB. Jika waktu download habis saat deployment, periksa kecepatan koneksi internet Anda dan coba lagi lakukan deployment.

Kelola perangkat inti Greengrass dengan AWS Systems Manager

> **Note**
>
> AWS IoT Greengrasssaat ini tidak mendukung fitur ini di perangkat inti Windows.

Systems Manager adalah AWS layanan yang dapat Anda gunakan untuk melihat dan mengontrol infrastrukturAWS, termasuk instans Amazon EC2, server lokal dan mesin virtual (VM), dan perangkat edge. Systems Manager memungkinkan Anda untuk melihat data operasional, mengotomatiskan tugas operasi, dan menjaga keamanan dan kepatuhan. Ketika Anda mendaftarkan mesin dengan Systems Manager, itu disebut node terkelola. Untuk informasi selengkapnya, lihat Apa itu AWS Systems Manager? dalam Panduan Pengguna AWS Systems Manager.

AWS Systems ManagerAgen (Systems Manager Agent) adalah perangkat lunak yang dapat Anda instal pada perangkat untuk memungkinkan Systems Manager memperbarui, mengelola, dan mengonfigurasinya. Untuk menginstal Agen Systems Manager pada perangkat inti Greengrass, gunakan komponen Systems Manager Agent. Saat Anda menggunakan Agen Systems Manager untuk pertama kalinya, itu akan mendaftarkan perangkat inti sebagai node terkelola Systems Manager. Agen Systems Manager berjalan pada perangkat untuk memungkinkan komunikasi dengan layanan Systems Manager diAWS Cloud. Untuk informasi selengkapnya tentang cara menginstal dan mengonfigurasi komponen Systems Manager Agent, lihatInstal AWS Systems Manager Agen.

Alat dan fitur Systems Manager disebut kemampuan. Perangkat inti Greengrass mendukung semua kemampuan Systems Manager. Untuk informasi selengkapnya tentang kemampuan ini dan cara menggunakan Systems Manager untuk mengelola perangkat inti, lihat kemampuan Systems Manager di Panduan AWS Systems Manager Pengguna.

AWS Systems Managermenawarkan tingkat instans standar dan tingkat instans lanjutan untuk node terkelola Systems Manager. Jika Anda menggunakan Systems Manager untuk pertama kalinya, Anda mulai pada tingkat instans standar. Pada tingkat instans standar, Anda dapat mendaftarkan hingga 1.000 node terkelola per node. Wilayah AWS Akun AWS Jika Anda perlu mendaftarkan lebih dari 1.000 node terkelola dalam satu akun dan Wilayah, atau jika Anda perlu menggunakan kemampuan Session Manager, gunakan tingkat instance lanjutan. Untuk informasi selengkapnya, lihat Mengonfigurasi tingkatan instance di AWS Systems ManagerPanduan Pengguna.

Topik

Instal AWS Systems Manager Agen

AWS Systems ManagerAgen (Systems Manager Agent) adalah perangkat lunak Amazon yang Anda instal untuk mengaktifkan Systems Manager memperbarui, mengelola, dan mengonfigurasi perangkat inti Greengrass, instans Amazon EC2, dan sumber daya lainnya. Agen memproses dan menjalankan permintaan dari layanan Systems Manager diAWS Cloud. Kemudian, agen mengirimkan informasi status dan runtime kembali ke layanan Systems Manager. Untuk informasi selengkapnya, lihat Tentang Agen Systems Manager di Panduan AWS Systems Manager Pengguna.

AWSmenyediakan Agen Systems Manager sebagai komponen Greengrass yang dapat Anda gunakan ke perangkat inti Greengrass Anda untuk mengelolanya dengan Systems Manager. Komponen Systems Manager Agent menginstal perangkat lunak Systems Manager Agent dan mendaftarkan perangkat inti sebagai node terkelola di Systems Manager. Ikuti langkah-langkah di halaman ini untuk menyelesaikan prasyarat dan menerapkan komponen Systems Manager Agent ke perangkat inti atau grup perangkat inti.

Topik

Langkah 1: Menyelesaikan langkah-langkah pengaturan umum Systems Manager

Jika Anda belum melakukannya, selesaikan langkah-langkah pengaturan umum untukAWS Systems Manager. Untuk informasi selengkapnya, lihat Lengkapi langkah-langkah penyiapan Systems Manager umum di Panduan AWS Systems Manager Pengguna.

Langkah 2: Buat peran layanan IAM untuk Systems Manager

Agen Systems Manager menggunakan peran layanan AWS Identity and Access Management (IAM) untuk berkomunikasi denganAWS Systems Manager. Systems Manager mengasumsikan peran ini untuk mengaktifkan kemampuan Systems Manager pada setiap perangkat inti. Komponen Systems Manager Agent juga menggunakan peran ini untuk mendaftarkan perangkat inti sebagai node terkelola Systems Manager saat Anda menerapkan komponen. Jika Anda belum melakukannya, buat peran layanan Systems Manager untuk komponen Systems Manager Agent yang akan digunakan. Untuk informasi selengkapnya, lihat Membuat peran layanan IAM untuk perangkat edge di Panduan AWS Systems Manager Pengguna.

Langkah 3: Tambahkan izin ke peran pertukaran token

Perangkat inti Greengrass menggunakan peran layanan IAM, yang disebut peran pertukaran token, untuk berinteraksi dengan layanan. AWS Setiap perangkat inti memiliki peran pertukaran token yang Anda buat saat Anda menginstal perangkat lunak AWS IoT Greengrass Core. Banyak komponen Greengrass, seperti Agen Systems Manager, memerlukan izin tambahan pada peran ini. Komponen agen Systems Manager memerlukan izin berikut, yang mencakup izin untuk menggunakan peran yang Anda buat. Langkah 2: Buat peran layanan IAM untuk Systems Manager

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::account-id:role/SSMServiceRole"
      ]
    },
    {
      "Action": [
        "ssm:AddTagsToResource",
        "ssm:RegisterManagedInstance"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
```

```
}
```

Jika Anda belum melakukannya, tambahkan izin ini ke peran pertukaran token perangkat inti untuk memungkinkan Agen Systems Manager beroperasi. Anda dapat menambahkan kebijakan baru ke peran pertukaran token untuk memberikan izin ini.

Untuk menambahkan izin ke peran pertukaran token (konsol)

1. Di menu navigasi konsol IAM, pilih Peran.
2. Pilih peran IAM yang Anda atur sebagai peran pertukaran token saat Anda menginstal perangkat lunak AWS IoT Greengrass Core. Jika Anda tidak menentukan nama untuk peran pertukaran token saat Anda menginstal perangkat lunak AWS IoT Greengrass Core, itu membuat peran bernama`GreengrassV2TokenExchangeRole`.
3. Di bawah Izin, pilih Tambahkan izin, lalu pilih Lampirkan kebijakan.
4. Pilih Buat kebijakan. Halaman Buat kebijakan terbuka di tab browser baru.
5. Pada halaman Buat kebijakan, lakukan hal berikut:

   a. Pilih JSON untuk membuka editor JSON.

   b. Tempelkan kebijakan berikut ke editor JSON. Ganti *`SSM ServiceRole`* dengan nama peran layanan yang Anda buat. Langkah 2: Buat peran layanan IAM untuk Systems Manager

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::account-id:role/SSMServiceRole"
      ]
    },
    {
      "Action": [
        "ssm:AddTagsToResource",
        "ssm:RegisterManagedInstance"
      ],
      "Effect": "Allow",
```

```
      "Resource": "*"
    }
  ]
}
```

c. Pilih Berikutnya: Tanda.

d. Pilih Berikutnya: Tinjauan.

e. Masukkan Nama untuk kebijakan, seperti**`GreengrassSSMAgentComponentPolicy`**.

f. Pilih Buat kebijakan.

g. Beralih ke tab browser sebelumnya di mana Anda memiliki peran pertukaran token terbuka.

6. Pada halaman Tambah izin, pilih tombol refresh, lalu pilih kebijakan agen Greengrass Systems Manager yang Anda buat di langkah sebelumnya.

7. Pilih Lampirkan kebijakan.

   Perangkat inti yang menggunakan peran pertukaran token ini sekarang memiliki izin untuk berinteraksi dengan layanan Systems Manager.

Untuk menambahkan izin ke peran pertukaran token () AWS CLI

Untuk menambahkan kebijakan yang memberikan izin untuk menggunakan Systems Manager

1. Buat file bernama `ssm-agent-component-policy.json` dan salin JSON berikut ke dalam file. Ganti *`SSM ServiceRole`* dengan nama peran layanan yang Anda buat. Langkah 2: Buat peran layanan IAM untuk Systems Manager

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::account-id:role/SSMServiceRole"
      ]
    },
    {
      "Action": [
```

```
        "ssm:AddTagsToResource",
        "ssm:RegisterManagedInstance"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

2. Jalankan perintah berikut untuk membuat kebijakan dari dokumen kebijakan di `ssm-agent-component-policy.json`.

   Linux or Unix

   ```
   aws iam create-policy \
     --policy-name GreengrassSSMAgentComponentPolicy \
     --policy-document file://ssm-agent-component-policy.json
   ```

   Windows Command Prompt (CMD)

   ```
   aws iam create-policy ^
     --policy-name GreengrassSSMAgentComponentPolicy ^
     --policy-document file://ssm-agent-component-policy.json
   ```

   PowerShell

   ```
   aws iam create-policy `
     --policy-name GreengrassSSMAgentComponentPolicy `
     --policy-document file://ssm-agent-component-policy.json
   ```

   Salin Amazon Resource Name (ARN) kebijakan dari metadata kebijakan dalam output. Anda menggunakan ARN ini untuk melampirkan kebijakan ini ke peran perangkat inti di langkah berikutnya.

3. Jalankan perintah berikut untuk melampirkan kebijakan ke peran pertukaran token.

   - Ganti *GreenGrassV2 TokenExchangeRole* dengan nama peran pertukaran token yang Anda tentukan saat Anda menginstal perangkat lunak Core. AWS IoT Greengrass Jika Anda tidak menentukan nama untuk peran pertukaran token saat

Anda menginstal perangkat lunak AWS IoT Greengrass Core, itu membuat peran bernama`GreengrassV2TokenExchangeRole`.

- Ganti ARN kebijakan dengan ARN dari langkah sebelumnya.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Jika perintah tidak memiliki output, itu berhasil. Perangkat inti yang menggunakan peran pertukaran token ini sekarang memiliki izin untuk berinteraksi dengan layanan Systems Manager.

Langkah 4: Menyebarkan komponen Systems Manager Agent

Selesaikan langkah-langkah berikut untuk menerapkan dan mengkonfigurasi komponen Systems Manager Agent. Anda dapat menyebarkan komponen ke perangkat inti tunggal atau ke sekelompok perangkat inti.

Untuk menyebarkan komponen Agen Systems Manager (konsol)

1. Pada menu navigasi konsol AWS IoT Greengrass tersebut, pilih Komponen.

2. Pada halaman Components, pilih tab Public components, lalu pilih aws.greengrass.SystemsManagerAgent.
3. Pada halaman aws.greengrass.SystemsManagerAgent pilih Deploy.
4. Dari Tambahkan ke penerapan, pilih penerapan yang ada untuk direvisi, atau pilih untuk membuat penerapan baru, lalu pilih Berikutnya.
5. Jika Anda memilih untuk membuat penerapan baru, pilih perangkat inti target atau grup hal untuk penerapan. Pada halaman Tentukan target, di bawah target Deployment, pilih perangkat inti atau grup benda, lalu pilih Berikutnya.
6. Pada halaman Pilih komponen, verifikasi bahwa aws.greengrass.SystemsManagerAgentkomponen dipilih, pilih Berikutnya.
7. Pada halaman Configure components, pilih aws.greengrass.SystemsManagerAgent, lalu lakukan hal berikut:

   a. Pilih Konfigurasi komponen.

   b. Dalam konfigurasi aws.greengrass.SystemsManagerAgent modal, di bawah Configuration update, di Configuration to merge, masukkan update konfigurasi berikut. Ganti *SSM ServiceRole* dengan nama peran layanan yang Anda buat. Langkah 2: Buat peran layanan IAM untuk Systems Manager

```
{
  "SSMRegistrationRole": "SSMServiceRole",
  "SSMOverrideExistingRegistration": false
}
```

> **Note**
>
> Jika perangkat inti sudah menjalankan Agen Systems Manager yang terdaftar dengan aktivasi hibrida, ubah `SSMOverrideExistingRegistration` ke`true`. Parameter ini menentukan apakah komponen Systems Manager Agent mendaftarkan perangkat inti saat Agen Systems Manager sudah berjalan di perangkat dengan aktivasi hybrid.
>
> Anda juga dapat menentukan tag (`SSMResourceTags`) untuk ditambahkan ke node terkelola Systems Manager yang dibuat oleh komponen Systems Manager Agent untuk perangkat inti. Untuk informasi selengkapnya, lihat konfigurasi komponen Agen Manajer Sistem.

c. Pilih Konfirmasi untuk menutup modal, lalu pilih Berikutnya.

8. Pada halaman Konfigurasikan pengaturan lanjutan, simpan pengaturan konfigurasi default tersebut, dan pilih Selanjutnya.
9. Di halaman Tinjau, pilih Deploy.

   Penyebaran dapat memakan waktu hingga satu menit untuk diselesaikan.

Untuk menyebarkan komponen Agen Systems Manager () AWS CLI

Untuk menerapkan komponen Systems Manager Agent, buat dokumen penerapan yang disertakan `aws.greengrass.SystemsManagerAgent` dalam `components` objek, dan tentukan pemutakhiran konfigurasi untuk komponen tersebut. Ikuti petunjuk Buat deployment untuk membuat penerapan baru atau merevisi penerapan yang ada.

Contoh berikut dokumen penyebaran sebagian menentukan untuk menggunakan peran layanan bernama. `SSMServiceRole` Ganti *`SSM ServiceRole`* dengan nama peran layanan yang Anda buat. Langkah 2: Buat peran layanan IAM untuk Systems Manager

```
{
  ...,
  "components": {
    ...,
    "aws.greengrass.SystemsManagerAgent": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"SSMRegistrationRole\":\"SSMServiceRole\",
\"SSMOverrideExistingRegistration\":false}"
      }
    }
  }
}
```

> **Note**
>
> Jika perangkat inti sudah menjalankan Agen Systems Manager yang terdaftar dengan aktivasi hibrida, ubah `SSMOverrideExistingRegistration` ke`true`. Parameter ini menentukan apakah komponen Systems Manager Agent mendaftarkan perangkat inti saat Agen Systems Manager sudah berjalan di perangkat dengan aktivasi hybrid.

Anda juga dapat menentukan tag (`SSMResourceTags`) untuk ditambahkan ke node terkelola Systems Manager yang dibuat oleh komponen Systems Manager Agent untuk perangkat inti. Untuk informasi selengkapnya, lihat konfigurasi komponen Agen Manajer Sistem.

Deployment ini dapat memakan waktu beberapa menit hingga selesai. Anda dapat menggunakan AWS IoT Greengrass layanan untuk memeriksa status penyebaran, dan Anda dapat memeriksa log perangkat lunak AWS IoT Greengrass inti dan log komponen Agen Systems Manager untuk memverifikasi bahwa Agen Manajer Sistem berjalan dengan sukses. Untuk informasi selengkapnya, lihat hal berikut:

- Periksa status deployment
- Memantau AWS IoT Greengrass log
- Melihat log Agen Systems Manager di Panduan AWS Systems Manager Pengguna

Jika penerapan gagal atau Agen Systems Manager tidak berjalan, Anda dapat memecahkan masalah penerapan pada setiap perangkat inti. Untuk informasi selengkapnya, lihat hal berikut:

- Pemecahan masalah AWS IoT Greengrass V2
- Pemecahan Masalah Agen Systems Manager di Panduan Pengguna AWS Systems Manager

Langkah 5: Verifikasi pendaftaran perangkat inti dengan Systems Manager

Ketika komponen Systems Manager Agent berjalan, ia mendaftarkan perangkat inti sebagai node terkelola di Systems Manager. Anda dapat menggunakan AWS IoT Greengrass konsol, konsol Systems Manager, dan Systems Manager API untuk memverifikasi bahwa perangkat inti terdaftar sebagai node terkelola. Node terkelola juga disebut instance di bagian AWS konsol dan API.

Untuk memverifikasi pendaftaran perangkat inti (AWS IoT Greengrasskonsol)

1. Di menu navigasi konsol AWS IoT Greengrass tersebut, pilih Perangkat inti.
2. Pilih perangkat inti untuk memverifikasi.
3. Pada halaman detail perangkat inti, temukan properti AWS Systems Managerinstance. Jika properti ini ada dan menampilkan link ke konsol Systems Manager, perangkat inti terdaftar sebagai node terkelola.

Anda juga dapat menemukan properti status AWS Systems Manager ping untuk memeriksa status Agen Systems Manager pada perangkat inti. Ketika statusnya Online, Anda dapat mengelola perangkat inti dengan Systems Manager.

Untuk memverifikasi pendaftaran perangkat inti (konsol Systems Manager)

1. Di menu navigasi konsol Systems Manager, pilih Fleet Manager.
2. Di bawah Node terkelola, lakukan hal berikut:
   a. Tambahkan filter di mana tipe Sumber berada AWS::IoT::Thing.
   b. Tambahkan filter di mana Source ID adalah nama perangkat inti untuk memverifikasi.
3. Temukan perangkat inti dalam tabel Managed nodes. Jika perangkat inti ada di tabel, itu terdaftar sebagai node terkelola.

   Anda juga dapat menemukan properti status ping Agen Manajer Systems untuk memeriksa status Agen Systems Manager pada perangkat inti. Ketika statusnya Online, Anda dapat mengelola perangkat inti dengan Systems Manager.

Untuk memverifikasi registrasi perangkat inti (AWS CLI)

- Gunakan DescribeInstanceInformationoperasi untuk mendapatkan daftar node terkelola yang cocok dengan filter yang Anda tentukan. Jalankan perintah berikut untuk memverifikasi apakah perangkat inti terdaftar sebagai node terkelola. Ganti *MyGreengrassCore*dengan nama perangkat inti untuk memverifikasi.

  ```
  aws ssm describe-instance-information --filter
   Key=SourceIds,Values=MyGreengrassCore Key=SourceTypes,Values=AWS::IoT::Thing
  ```

  Respons berisi daftar node terkelola yang cocok dengan filter. Jika daftar berisi node terkelola, perangkat inti terdaftar sebagai node terkelola. Anda juga dapat menemukan informasi lain tentang node terkelola perangkat inti dalam respons. Jika `PingStatus` propertinya`Online`, Anda dapat mengelola perangkat inti dengan Systems Manager.

Setelah memverifikasi bahwa perangkat inti terdaftar sebagai node terkelola di Systems Manager, Anda dapat menggunakan konsol Systems Manager dan API untuk mengelola perangkat inti tersebut. Untuk informasi selengkapnya tentang kemampuan Systems Manager yang dapat Anda

gunakan untuk mengelola perangkat inti Greengrass, lihat kemampuan Systems Manager di Panduan Pengguna. AWS Systems Manager

Menghapus instalasiAWS Systems Manager agen

Jika Anda tidak lagi ingin mengelola perangkat inti GreengrassAWS Systems Manager, Anda dapat membatalkan pendaftaran perangkat inti dari Systems Manager dan menghapus instalanAWS Systems Manager Agen (Agen Systems Manager) dari perangkat.

Anda dapat melakukan instalasi perangkat inti kembali kapan saja. Untuk melakukannya, gunakan komponen Agen Systems Manager lagi, yang mendaftarkan perangkat inti dengan Systems Manager saat menginstal. Systems Manager menyimpan riwayat perintah untuk perangkat inti yang tidak terdaftar selama 30 hari.

Topik

Langkah 1: Deregister perangkat inti dari Systems Manager

Anda dapat menggunakan konsol Systems Manager atau API untuk membatalkan instalasi perangkat inti. Untuk informasi selengkapnya, lihat Membatalkan pendaftaran node terkelola di PanduanAWS Systems Manager Pengguna.

Langkah 2: Copot komponen Agen Systems Manager

Setelah Anda membatalkan pendaftaran perangkat inti, hapus instalan komponen Agen Systems Manager dari perangkat. Untuk menghapus komponen dari perangkat inti Greengrass, revisi penyebaran yang menginstal komponen, dan menghapus komponen dari penyebaran. Perangkat lunakAWS IoT Greengrass Core menghapus komponen ketika tidak ada penerapan perangkat inti yang menentukan komponen itu. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat.

Untuk menghapus komponen Agen Systems Manager (konsol)

1. Di menu navigasi konsol AWS IoT Greengrass tersebut, pilih Perangkat inti.

2. Pilih perangkat inti tempat Anda ingin menghapus komponen Agen Systems Manager.
3. Pada halaman detail perangkat inti, pilih tab Deployment.
4. Pilih penyebaran yang menyebarkan komponen Agen Systems Manager ke perangkat inti.
5. Pada halaman detail penyebaran, pilih Revisi.
6. Dalam Merevisi modal penyebaran, pilih Merevisi penyebaran.
7. Di Langkah 1: Tentukan target, pilih Berikutnya.
8. Pada Langkah 2: Pilih komponen, kosongkan seleksi untuk aws.greengrass.SystemsManagerAgentkomponen, lalu pilih Berikutnya.
9. Pada Langkah 3: Konfigurasikan komponen, pilih Berikutnya.
10. Di Langkah 4: Konfigurasikan pengaturan lanjutan, pilih Berikutnya.
11. Pada Langkah 5: Tinjau, pilih Deploy.

Untuk menghapus komponen Agen Systems Manager (CLI)

Untuk menghapus komponen Agen Systems Manager, revisi penyebaran yang menerapkannya, dan menghapusnya dari penyebaran. Untuk informasi selengkapnya, lihat Revisi deployment.

Deployment ini dapat memakan waktu beberapa menit hingga selesai. Anda dapat menggunakanAWS IoT Greengrass layanan untuk memeriksa status deployment. Untuk informasi selengkapnya, lihat Periksa status deployment.

Langkah 3: Uninstall perangkat lunak Systems Manager agen

Perangkat lunak Agen Systems Manager terus berjalan di perangkat inti setelah Anda menghapus komponen Agen Systems Manager. Untuk menghapus perangkat agen Systems Manager, Anda dapat menjalankan perintah pada perangkat inti. Untuk informasi selengkapnya, lihat Menghapus Agen Systems Manager dari instans Linux di PanduanAWS Systems Manager Pengguna.

Keamanan di AWS IoT Greengrass

Keamanan cloud di AWS merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda 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 menjelaskan hal ini sebagai keamanan dari cloud dan keamanan dalam cloud:

- Keamanan cloud – AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan layanan-layanan AWS di AWS Cloud Cloud. AWS juga memberikan Anda layanan yang dapat digunakan 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 berlakuAWS IoT Greengrass, lihat AWSLayanan dalam Lingkup oleh AWS Layanan Program Kepatuhan .
- 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

- Keamanan infrastruktur dalam AWS IoT Greengrass
- Analisis konfigurasi dan kerentanan dalam AWS IoT Greengrass
- Integritas kode diAWS IoT Greengrass V2
- AWS IoT Greengrass dan titik akhir VPC antarmuka (AWS PrivateLink)
- Praktik terbaik keamanan untuk AWS IoT Greengrass

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

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

AWS IoT Greengrass memiliki dua mode komunikasi di mana data dalam transit:

- the section called “Data dalam transit melalui internet”. Komunikasi antara inti Greengrass dan AWS IoT Greengrass melalui internet dienkripsi.

- the section called “Data pada perangkat inti”. 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 selengkapnya, lihat Keamanan transportasi di Panduan Developer AWS IoT Core.

Data pada perangkat inti

AWS IoT Greengrass tidak mengenkripsi data yang dipertukarkan secara lokal pada perangkat inti Greengrass karena data tidak meninggalkan perangkat. Ini termasuk komunikasi antar komponen yang ditetapkan pengguna, AWS IoT SDK perangkat, dan komponen publik, seperti pengelola stream.

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 komponen secret manager.

Manajemen kunci untuk perangkat inti Greengrass

Adalah tanggung jawab pelanggan untuk menjamin penyimpanan kunci kriptografi yang aman (publik dan privat) pada perangkat inti 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 inti 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.

Sebuah perangkat inti Greengrass mendukung penyimpanan kunci privat menggunakan izin sistem file ataumodul keamanan perangkat keras. Jika Anda menggunakan kunci privat berbasis sistem file, Anda bertanggung jawab atas penyimpanannya yang aman pada perangkat inti.

Integrasi keamanan perangkat keras

> **Note**
>
> Fitur ini tersedia untuk v2.5.3 dan yang lebih baru dari komponen inti Greengrass. AWS IoT Greengrasssaat ini tidak mendukung fitur ini di perangkat inti Windows.

Anda dapat mengonfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan modul keamanan perangkat keras (HSM) melalui antarmuka PKCS #11. Fitur ini memungkinkan Anda menyimpan kunci pribadi dan sertifikat perangkat dengan aman sehingga tidak terekspos atau digandakan dalam perangkat lunak. Anda dapat menyimpan kunci pribadi dan sertifikat pada modul perangkat keras seperti HSM atau Trusted Platform Module (TPM).

Perangkat lunak AWS IoT Greengrass Core menggunakan kunci pribadi dan sertifikat X.509 untuk mengautentikasi koneksi ke dan layanan. AWS IoT AWS IoT Greengrass Komponen manajer rahasia menggunakan kunci pribadi ini untuk mengenkripsi dan mendekripsi rahasia yang Anda terapkan ke perangkat inti Greengrass dengan aman. Saat Anda mengonfigurasi perangkat inti untuk menggunakan HSM, komponen ini menggunakan kunci pribadi dan sertifikat yang Anda simpan di HSM.

Komponen broker Moquette MQTT juga menyimpan kunci pribadi untuk sertifikat server MQTT lokalnya. Komponen ini menyimpan kunci pribadi pada sistem file perangkat di folder kerja komponen. Saat ini, AWS IoT Greengrass tidak mendukung penyimpanan kunci pribadi atau sertifikat ini di HSM.

> **Tip**
>
> Cari perangkat yang mendukung fitur ini di Katalog Perangkat AWS Mitra.

Topik

Persyaratan

Anda harus memenuhi persyaratan berikut untuk menggunakan HSM pada perangkat inti Greengrass:

- Greengrass nucleus v2.5.3 atau yang lebih baru diinstal pada perangkat inti. Anda dapat memilih versi yang kompatibel saat menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti.
- Komponen penyedia PKCS #11 diinstal pada perangkat inti. Anda dapat mengunduh dan menginstal komponen ini ketika Anda menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat inti.
- Modul keamanan perangkat keras yang mendukung skema tanda tangan PKCS #1 v1.5 dan kunci RSA dengan ukuran kunci RSA-2048 (atau lebih besar) atau kunci ECC.

> **Note**
>
> Untuk menggunakan modul keamanan perangkat keras dengan kunci ECC, Anda harus menggunakan Greengrass nucleus v2.5.6 atau yang lebih baru.

Untuk menggunakan modul keamanan perangkat keras dan manajer rahasia, Anda harus menggunakan modul keamanan perangkat keras dengan kunci RSA.

- Pustaka penyedia PKCS #11 yang dapat dimuat oleh perangkat lunak AWS IoT Greengrass Core saat runtime (menggunakan libdl) untuk menjalankan fungsi PKCS #11. Pustaka penyedia PKCS #11 harus mengimplementasikan operasi API PKCS #11 berikut:
  - `C_Initialize`
  - `C_Finalize`
  - `C_GetSlotList`
  - `C_GetSlotInfo`
  - `C_GetTokenInfo`
  - `C_OpenSession`
  - `C_GetSessionInfo`
  - `C_CloseSession`
  - `C_Login`
  - `C_Logout`
  - `C_GetAttributeValue`
  - `C_FindObjectsInit`
  - `C_FindObjects`
  - `C_FindObjectsFinal`
  - `C_DecryptInit`
  - `C_Decrypt`
  - `C_DecryptUpdate`
  - `C_DecryptFinal`
  - `C_SignInit`
  - `C_Sign`
  - `C_SignUpdate`
  - `C_SignFinal`
  - `C_GetMechanismList`
  - `C_GetMechanismInfo`
  - `C_GetInfo`

  - `C_GetFunctionList`
- Modul perangkat keras harus dapat diatasi dengan label slot, sebagaimana ditentukan di dalam spesifikasi PKCS#11.
- Anda harus menyimpan kunci pribadi dan sertifikat di HSM di slot yang sama, dan mereka harus menggunakan label objek dan ID objek yang sama, jika HSM mendukung ID objek.
- Sertifikat dan kunci pribadi harus dapat diselesaikan dengan label objek.
- Kunci pribadi harus memiliki izin berikut:
  - `sign`
  - `decrypt`
- (Opsional) Untuk menggunakan komponen manajer rahasia, Anda harus menggunakan versi 2.1.0 atau yang lebih baru, dan kunci pribadi harus memiliki izin berikut:
  - `unwrap`
  - `wrap`

Praktik terbaik keamanan perangkat keras

Pertimbangkan praktik terbaik berikut saat Anda mengonfigurasi keamanan perangkat keras pada perangkat inti Greengrass.

- Hasilkan kunci privat langsung pada HSM dengan menggunakan generator nomor acak perangkat keras internal. Pendekatan ini lebih aman daripada mengimpor kunci pribadi yang Anda hasilkan di tempat lain, karena kunci pribadi tetap berada dalam HSM.
- Konfigurasikan kunci pribadi agar tidak dapat diubah dan melarang ekspor.
- Gunakan alat penyediaan yang direkomendasikan oleh vendor perangkat keras HSM untuk membuat permintaan penandatanganan sertifikat (CSR) menggunakan kunci pribadi yang dilindungi perangkat keras, lalu gunakan konsol atau API untuk menghasilkan sertifikat klien. AWS IoT

> **Note**
>
> Praktik keamanan terbaik untuk memutar kunci tidak berlaku saat Anda membuat kunci pribadi pada HSM.

Instal perangkat lunak AWS IoT Greengrass Core dengan keamanan perangkat keras

Saat Anda menginstal perangkat lunak AWS IoT Greengrass Core, Anda dapat mengonfigurasinya untuk menggunakan kunci pribadi yang Anda hasilkan di HSM. Pendekatan ini mengikuti praktik terbaik keamanan untuk menghasilkan kunci pribadi di HSM, sehingga kunci pribadi tetap berada dalam HSM.

Untuk menginstal perangkat lunak AWS IoT Greengrass Core dengan keamanan perangkat keras, Anda melakukan hal berikut:

1. Hasilkan kunci pribadi di HSM.
2. Buat permintaan penandatanganan sertifikat (CSR) dari kunci pribadi.
3. Buat sertifikat dari CSR. Anda dapat membuat sertifikat yang ditandatangani oleh AWS IoT atau oleh otoritas sertifikat root (CA) lainnya. Untuk informasi selengkapnya tentang cara menggunakan CA root lain, lihat Membuat sertifikat klien Anda sendiri di Panduan AWS IoT Core Pengembang.
4. Unduh AWS IoT sertifikat dan impor ke HSM.
5. Instal perangkat lunak AWS IoT Greengrass Core dari file konfigurasi yang menentukan untuk menggunakan komponen penyedia PKCS #11 dan kunci pribadi dan sertifikat di HSM.

Anda dapat memilih salah satu opsi instalasi berikut untuk menginstal perangkat lunak AWS IoT Greengrass Core dengan keamanan perangkat keras:

- Instalasi manual

  Pilih opsi ini untuk membuat AWS sumber daya yang diperlukan secara manual dan mengkonfigurasi keamanan perangkat keras. Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya manual.
- Instalasi dengan penyediaan khusus

  Pilih opsi ini untuk mengembangkan aplikasi Java khusus yang secara otomatis membuat AWS sumber daya yang diperlukan dan mengonfigurasi keamanan perangkat keras. Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya khusus.

Saat ini, AWS IoT Greengrass tidak mendukung penginstalan perangkat lunak AWS IoT Greengrass Core dengan keamanan perangkat keras saat Anda menginstal dengan penyediaan sumber daya otomatis atau penyediaan AWS IoT armada.

Konfigurasikan keamanan perangkat keras pada perangkat inti yang ada

Anda dapat mengimpor kunci pribadi dan sertifikat perangkat inti ke HSM untuk mengonfigurasi keamanan perangkat keras.

> **Pertimbangan**
>
> - Anda harus memiliki akses root ke sistem file perangkat inti.
> - Dalam prosedur ini, Anda mematikan perangkat lunak AWS IoT Greengrass Core, sehingga perangkat inti offline dan tidak tersedia saat Anda mengonfigurasi keamanan perangkat keras.

Untuk mengonfigurasi keamanan perangkat keras pada perangkat inti yang ada, Anda melakukan hal berikut:

1. Inisialisasi HSM.
2. Terapkan komponen penyedia PKCS #11 ke perangkat inti.
3. Hentikan perangkat lunak AWS IoT Greengrass inti.
4. Impor kunci pribadi perangkat inti dan sertifikat ke HSM.
5. Perbarui file konfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan kunci pribadi dan sertifikat di HSM.
6. Mulai perangkat lunak AWS IoT Greengrass inti.

Langkah 1: Inisialisasi modul keamanan perangkat keras

Selesaikan langkah berikut untuk menginisialisasi HSM pada perangkat inti Anda.

Untuk menginisialisasi modul keamanan perangkat keras

- Inisialisasi token PKCS #11 di HSM, dan simpan ID slot dan PIN pengguna untuk token tersebut. Periksa dokumentasi untuk HSM Anda untuk mempelajari cara menginisialisasi token. Anda menggunakan ID slot dan PIN pengguna nanti saat Anda menerapkan dan mengonfigurasi komponen penyedia PKCS #11.

Langkah 2: Terapkan komponen penyedia PKCS #11

Selesaikan langkah-langkah berikut untuk menerapkan dan mengonfigurasi komponen penyedia PKCS #11. Anda dapat menerapkan komponen ke satu atau beberapa perangkat inti.

Untuk menerapkan komponen penyedia PKCS #11 (konsol)

1. Pada menu navigasi konsol AWS IoT Greengrass tersebut, pilih Komponen.
2. Pada halaman Components, pilih tab Public components, lalu pilih aws.greengrass.crypto.Pkcs11Provider.
3. Pada halaman aws.greengrass.crypto.Pkcs11Provider pilih Deploy.
4. Dari Tambahkan ke penerapan, pilih penerapan yang ada untuk direvisi, atau pilih untuk membuat penerapan baru, lalu pilih Berikutnya.
5. Jika Anda memilih untuk membuat penerapan baru, pilih perangkat inti target atau grup hal untuk penerapan. Pada halaman Tentukan target, di bawah target Deployment, pilih perangkat inti atau grup benda, lalu pilih Berikutnya.
6. Pada halaman Pilih komponen, di bawah Komponen publik, pilih aws.greengrass.crypto.Pkcs11Provider, lalu pilih Berikutnya.
7. Pada halaman Configure components, pilih aws.greengrass.crypto.Pkcs11Provider, lalu lakukan hal berikut:

   a. Pilih Konfigurasi komponen.

   b. Dalam konfigurasi aws.greengrass.crypto.Pkcs11Provider modal, di bawah Configuration update, di Configuration to merge, masukkan update konfigurasi berikut. Perbarui parameter konfigurasi berikut dengan nilai untuk perangkat inti target. Tentukan ID slot dan PIN pengguna tempat Anda menginisialisasi token PKCS #11 sebelumnya. Anda mengimpor kunci pribadi dan sertifikat ke slot ini di HSM nanti.

      `name`

      Nama untuk konfigurasi PKCS #11.

      `library`

      Jalur file absolut ke pustaka implementasi PKCS #11 yang dapat dimuat oleh perangkat lunak AWS IoT Greengrass Core dengan libdl.

`slot`

ID slot yang berisi kunci pribadi dan sertifikat perangkat. Nilai ini berbeda dari indeks slot atau label slot.

`userPin`

PIN pengguna yang digunakan untuk mengakses slot.

```
{
  "name": "softhsm_pkcs11",
  "library": "/usr/lib/softhsm/libsofthsm2.so",
  "slot": 1,
  "userPin": "1234"
}
```

c. Pilih Konfirmasi untuk menutup modal, lalu pilih Berikutnya.

8. Pada halaman Konfigurasikan pengaturan lanjutan, simpan pengaturan konfigurasi default tersebut, dan pilih Selanjutnya.
9. Di halaman Tinjau, pilih Deploy.

   Penyebaran dapat memakan waktu hingga satu menit untuk diselesaikan.

Untuk menerapkan komponen penyedia PKCS #11 () AWS CLI

Untuk menerapkan komponen penyedia PKCS #11, buat dokumen penerapan yang disertakan `aws.greengrass.crypto.Pkcs11Provider` dalam `components` objek, dan tentukan pemutakhiran konfigurasi untuk komponen tersebut. Ikuti petunjuk Buat deployment untuk membuat penerapan baru atau merevisi penerapan yang ada.

Contoh dokumen penerapan sebagian berikut menetapkan untuk menyebarkan dan mengkonfigurasi komponen penyedia PKCS #11. Perbarui parameter konfigurasi berikut dengan nilai untuk perangkat inti target. Simpan ID slot dan PIN pengguna untuk digunakan nanti saat Anda mengimpor kunci pribadi dan sertifikat ke HSM.

`name`

Nama untuk konfigurasi PKCS #11.

`library`

Jalur file absolut ke pustaka implementasi PKCS #11 yang dapat dimuat oleh perangkat lunak AWS IoT Greengrass Core dengan libdl.

`slot`

ID slot yang berisi kunci pribadi dan sertifikat perangkat. Nilai ini berbeda dari indeks slot atau label slot.

`userPin`

PIN pengguna yang digunakan untuk mengakses slot.

```
{
  "name": "softhsm_pkcs11",
  "library": "/usr/lib/softhsm/libsofthsm2.so",
  "slot": 1,
  "userPin": "1234"
}
```

```
{
  ...,
  "components": {
    ...,
    "aws.greengrass.crypto.Pkcs11Provider": {
      "componentVersion": "2.0.0",
      "configurationUpdate": {
        "merge": "{\"name\":\"softhsm_pkcs11\",\"library\":\"/usr/lib/softhsm/
libsofthsm2.so\",\"slot\":1,\"userPin\":\"1234\"}"
      }
    }
  }
}
```

Deployment ini dapat memakan waktu beberapa menit hingga selesai. Anda dapat menggunakan AWS IoT Greengrass layanan untuk memeriksa status penyebaran. Anda dapat memeriksa log perangkat lunak AWS IoT Greengrass Inti untuk memverifikasi bahwa komponen penyedia PKCS #11 berhasil diterapkan. Untuk informasi selengkapnya, lihat hal berikut:

- Periksa status deployment
- Memantau AWS IoT Greengrass log

Jika penerapan gagal, Anda dapat memecahkan masalah penerapan pada setiap perangkat inti. Untuk informasi selengkapnya, lihat Pemecahan masalah AWS IoT Greengrass V2.

Langkah 3: Perbarui konfigurasi pada perangkat inti

Perangkat lunak AWS IoT Greengrass Core menggunakan file konfigurasi yang menentukan bagaimana perangkat beroperasi. File konfigurasi ini mencakup tempat menemukan kunci pribadi dan sertifikat yang digunakan perangkat untuk terhubung ke fileAWS Cloud. Selesaikan langkah-langkah berikut untuk mengimpor kunci pribadi perangkat inti dan sertifikat ke HSM dan memperbarui file konfigurasi untuk menggunakan HSM.

Untuk memperbarui konfigurasi pada perangkat inti untuk menggunakan keamanan perangkat keras

1. Hentikan perangkat lunak AWS IoT Greengrass inti. Jika Anda mengkonfigurasi perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem dengan systemd, Anda dapat menjalankan perintah berikut untuk menghentikan perangkat lunak.

   ```
   sudo systemctl stop greengrass.service
   ```

2. Temukan kunci pribadi perangkat inti dan file sertifikat.

   - Jika Anda menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan otomatis atau penyediaan armada, kunci pribadi ada di*/greengrass/v2*/privKey.key, dan sertifikat ada di. */greengrass/v2*/thingCert.crt
   - Jika Anda menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan manual, kunci pribadi ada di secara */greengrass/v2*/private.pem.key default, dan sertifikat ada di secara */greengrass/v2*/device.pem.crt default.

   Anda juga dapat memeriksa system.privateKeyPath dan system.certificateFilePath properti di */greengrass/v2*/config/effectiveConfig.yaml untuk menemukan lokasi file-file ini.

3. Impor kunci pribadi dan sertifikat ke HSM. Periksa dokumentasi untuk HSM Anda untuk mempelajari cara mengimpor kunci pribadi dan sertifikat ke dalamnya. Impor kunci pribadi dan sertifikat menggunakan ID slot dan PIN pengguna tempat Anda menginisialisasi token PKCS #11 sebelumnya. Anda harus menggunakan label objek dan ID objek yang sama untuk kunci pribadi dan sertifikat. Simpan label objek yang Anda tentukan saat Anda mengimpor setiap file. Anda menggunakan label ini nanti ketika Anda memperbarui konfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan kunci pribadi dan sertifikat di HSM.

4. Perbarui konfigurasi AWS IoT Greengrass Core untuk menggunakan kunci pribadi dan sertifikat di HSM. Untuk memperbarui konfigurasi, Anda memodifikasi file konfigurasi AWS IoT Greengrass Core dan menjalankan perangkat lunak AWS IoT Greengrass Core dengan file konfigurasi yang diperbarui untuk menerapkan konfigurasi baru.

   Lakukan hal-hal berikut:

   a. Buat cadangan file konfigurasi AWS IoT Greengrass Core. Anda dapat menggunakan cadangan ini untuk memulihkan perangkat inti jika Anda mengalami masalah saat mengonfigurasi keamanan perangkat keras.

      ```
      sudo cp /greengrass/v2/config/effectiveConfig.yaml ~/ggc-config-backup.yaml
      ```

   b. Buka file konfigurasi AWS IoT Greengrass Core di editor teks. Misalnya, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk mengedit file. Ganti */greengrass/v2* dengan jalur ke folder root Greengrass.

      ```
      sudo nano /greengrass/v2/config/effectiveConfig.yaml
      ```

   c. Ganti nilai `system.privateKeyPath` dengan PKCS #11 URI untuk kunci pribadi di HSM. Ganti *iotdevicekey* dengan label objek tempat Anda mengimpor kunci pribadi dan sertifikat sebelumnya.

      ```
      pkcs11:object=iotdevicekey;type=private
      ```

   d. Ganti nilai `system.certificateFilePath` dengan PKCS #11 URI untuk sertifikat di HSM. Ganti *iotdevicekey* dengan label objek tempat Anda mengimpor kunci pribadi dan sertifikat sebelumnya.

      ```
      pkcs11:object=iotdevicekey;type=cert
      ```

   Setelah Anda menyelesaikan langkah-langkah ini, `system` properti dalam file konfigurasi AWS IoT Greengrass Core akan terlihat mirip dengan contoh berikut.

   ```
   system:
     certificateFilePath: "pkcs11:object=iotdevicekey;type=cert"
     privateKeyPath: "pkcs11:object=iotdevicekey;type=private"
     rootCaPath: "/greengrass/v2/rootCA.pem"
     rootpath: "/greengrass/v2"
   ```

```
  thingName: "MyGreengrassCore"
```

5. Terapkan konfigurasi dalam `effectiveConfig.yaml` file yang diperbarui. Jalankan `Greengrass.jar` dengan `--init-config` parameter untuk menerapkan konfigurasi dieffectiveConfig.yaml. Ganti */greengrass/v2* dengan jalur ke folder root Greengrass.

```
sudo java -Droot="/greengrass/v2" \
  -jar /greengrass/v2/alts/current/distro/lib/Greengrass.jar \
  --start false \
  --init-config /greengrass/v2/config/effectiveConfig.yaml
```

6. Mulai perangkat lunak AWS IoT Greengrass inti. Jika Anda mengkonfigurasi perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem dengan systemd, Anda dapat menjalankan perintah berikut untuk memulai perangkat lunak.

```
sudo systemctl start greengrass.service
```

   Untuk informasi selengkapnya, lihat Jalankan perangkat lunak inti AWS IoT Greengrass.

7. Periksa log perangkat lunak AWS IoT Greengrass inti untuk memverifikasi bahwa perangkat lunak dimulai dan terhubung ke fileAWS Cloud. Perangkat lunak AWS IoT Greengrass Core menggunakan kunci pribadi dan sertifikat untuk terhubung ke AWS IoT dan AWS IoT Greengrass layanan.

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

   Pesan log tingkat Info berikut menunjukkan bahwa perangkat lunak AWS IoT Greengrass Core berhasil terhubung ke AWS IoT dan AWS IoT Greengrass layanan.

```
2021-12-06T22:47:53.702Z [INFO] (Thread-3)
 com.aws.greengrass.mqttclient.AwsIotMqttClient: Successfully connected to AWS IoT
 Core. {clientId=MyGreengrassCore5, sessionPresent=false}
```

8. (Opsional) Setelah Anda memverifikasi bahwa perangkat lunak AWS IoT Greengrass Core berfungsi dengan kunci pribadi dan sertifikat di HSM, hapus kunci pribadi dan file sertifikat dari sistem file perangkat. Jalankan perintah berikut, dan ganti jalur file dengan jalur ke kunci pribadi dan file sertifikat.

```
sudo rm /greengrass/v2/privKey.key
```

```
sudo rm /greengrass/v2/thingCert.crt
```

Gunakan perangkat keras tanpa dukungan 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 pustaka PKCS #11 yang sesuai, atau jika Anda ingin menulis penyedia PKCS #11 kustom, hubungi perwakilan Amazon Web Services Enterprise Support Anda dengan pertanyaan terkait integrasi.

Lihat juga

- Panduan Penggunaan Antarmuka Token Kriptografi PKCS #11 Versi 2.4.0
- 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:

- Perangkat inti Greengrass

Perangkat inti Greengrass menggunakan sertifikat dan kebijakan AWS IoT untuk terhubung ke AWS IoT Core dan AWS IoT Greengrass. Sertifikat dan kebijakan juga memungkinkan AWS IoT Greengrass untuk menyebarkan komponen dan konfigurasi ke perangkat inti.

- Perangkat klien

  Perangkat klien MQTT menggunakan sertifikat dan kebijakan untuk terhubung ke AWS IoT Core dan layanan AWS IoT Greengrass. Hal ini memungkinkan perangkat klien untuk menggunakan penemuan cloud AWS IoT Greengrass untuk menemukan dan terhubung ke perangkat inti Greengrass. Perangkat klien menggunakan sertifikat yang sama untuk terhubung ke layanan cloud AWS IoT Core dan perangkat inti. Perangkat klien juga menggunakan informasi penemuan untuk autentikasi bersama dengan perangkat inti. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Sertifikat X.509

Komunikasi antara perangkat inti dan perangkat 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 lingkungan AWS IoT Greengrass, perangkat menggunakan sertifikat dengan kunci publik dan privat untuk koneksi Keamanan Lapisan Pengangkutan (TLS) berikut:

- Komponen klien AWS IoT pada perangkat inti Greengrass yang terhubung ke AWS IoT Core dan AWS IoT Greengrass melalui internet.
- Perangkat klien yang tersambung ke AWS IoT Greengrass melalui internet untuk menemukan perangkat inti.
- Komponen broker MQTT pada inti Greengrass menghubungkan ke perangkat Greengrass dalam kelompok melalui jaringan lokal.

Perangkat inti AWS IoT Greengrass menyimpan sertifikat dalam folder akar Greengrass.

Sertifikat otoritas sertifikasi (CA)

Perangkat inti Greengrass dan perangkat klien mengunduh sertifikat CA akar yang digunakan untuk autentikasi dengan layanan AWS IoT Core dan AWS IoT Greengrass. 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.

Perangkat klien juga mengunduh sertifikat CA perangkat inti Greengrass. Mereka menggunakan sertifikat ini untuk memvalidasi sertifikat server MQTT pada perangkat inti selama autentikasi bersama.

Rotasi sertifikat pada broker MQTT lokal

Saat Anda mengaktifkan dukungan perangkat klien, perangkat inti Greengrass menghasilkan sertifikat server MQTT lokal yang digunakan perangkat klien untuk otentikasi timbal balik. Sertifikat ini ditandatangani oleh sertifikat CA perangkat inti, yang disimpan oleh perangkat inti di cloud AWS IoT Greengrass. Perangkat klien mengambil sertifikat CA perangkat inti ketika mereka menemukan perangkat inti. Mereka menggunakan sertifikat CA perangkat inti untuk memverifikasi sertifikat server MQTT perangkat inti ketika mereka terhubung ke perangkat inti. Sertifikat CA perangkat inti berakhir setelah 5 tahun.

Sertifikat server MQTT kedaluwarsa setiap 7 hari secara default, dan Anda dapat mengonfigurasi durasi ini menjadi antara 2 dan 10 hari. Periode terbatas ini didasarkan pada praktik keamanan terbaik. Rotasi ini membantu mengurangi ancaman penyerang yang mencuri sertifikat server MQTT dan kunci privat untuk meniru perangkat inti Greengrass.

Perangkat inti Greengrass memutar sertifikat server MQTT 24 jam sebelum kedaluwarsa. Perangkat inti Greengrass menghasilkan sertifikat baru dan memulai ulang broker MQTT lokal. Ketika ini terjadi, semua perangkat klien yang terhubung ke perangkat inti Greengrass terputus. Perangkat klien dapat menyambung kembali ke perangkat inti Greengrass setelah waktu yang singkat.

Kebijakan AWS IoT untuk operasi bidang data

Gunakan kebijakan AWS IoT untuk mengotorisasi akses ke bidang data AWS IoT Core dan AWS IoT Greengrass. Bidang data AWS IoT Core menyediakan operasi untuk perangkat, pengguna, dan aplikasi. Operasi ini mencakup kemampuan untuk terhubung ke AWS IoT Core dan berlangganan topik. Bidang data AWS IoT Greengrass menyediakan operasi untuk perangkat Greengrass. Untuk informasi selengkapnya, lihat AWS IoT Greengrass V2 tindakan kebijakan. Operasi ini mencakup kemampuan untuk menyelesaikan dependensi komponen dan mengunduh artefak komponen publik.

Kebijakan AWS IoT adalah 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 selengkapnya, lihat kebijakan AWS IoT dan tindakan kebijakan AWS IoT di Panduan Developer AWS IoT Core.

> **Important**
>
> Variabel kebijakan objek (`iot:Connection.Thing.*`) tidak didukung untuk kebijakan AWS IoT untuk perangkat inti atau operasi bidang data Greengrass. Sebaliknya, Anda dapat menggunakan wildcard yang cocok dengan beberapa perangkat yang memiliki nama yang sama. Misalnya, Anda dapat menentukan `MyGreengrassDevice*` untuk cocok dengan `MyGreengrassDevice1`, `MyGreengrassDevice2`, dan sebagainya.

> **Note**
>
> AWS IoT Core memungkinkan Anda untuk melampirkan kebijakan AWS IoT pada grup sesuatu 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 V2 tindakan kebijakan

AWS IoT Greengrass V2mendefinisikan tindakan kebijakan berikut yang dapat digunakan perangkat inti Greengrass dan perangkat klien dalam kebijakan. AWS IoT Untuk menentukan sumber daya untuk tindakan kebijakan, Anda menggunakan Nama Sumber Daya Amazon (ARN) sumber daya.

Tindakan perangkat inti

`greengrass:GetComponentVersionArtifact`

Memberikan izin untuk mendapatkan URL yang telah ditetapkan sebelumnya untuk mengunduh artefak komponen publik atau artefak komponen Lambda.

Izin ini dievaluasi ketika perangkat inti menerima penerapan yang menentukan komponen publik atau Lambda yang memiliki artefak. Jika perangkat inti sudah memiliki artefak, ia tidak menngunduh artefak lagi.

Jenis sumber daya: `componentVersion`

Format ARN sumber daya: `arn:aws:greengrass:`*`region`*`:`*`account-id`*`:components:`*`component-name`*`:versions:`*`component-version`*

`greengrass:ResolveComponentCandidates`

Memberikan izin untuk mengidentifikasi daftar komponen yang memenuhi persyaratan komponen, versi, dan platform untuk deployment. Jika persyaratan tersebut bertentangan, atau tidak ada komponen yang memenuhi persyaratan, operasi ini akan kembali salah dan deployment tersebut gagal pada perangkat itu.

Izin ini dievaluasi ketika perangkat inti menerima deployment yang menentukan komponen publik yang memiliki artefak.

Jenis sumber daya: Tidak ada

Format ARN sumber daya: *

`greengrass:GetDeploymentConfiguration`

Memberikan izin untuk mendapatkan URL yang telah ditunjuk untuk mengunduh suatu dokumen deployment yang besar.

Izin ini dievaluasi ketika perangkat inti menerima deployment yang menentukan deployment dokumen yang lebih besar dari 7 KB (jika deployment tersebut menargetkan suatu objek) atau 31 KB (jika deployment tersebut menargetkan suatu grup objek). Dokumen deployment ini meliputi konfigurasi komponen, kebijakan deployment, dan metadata deployment. Untuk informasi selengkapnya, lihat Deploy komponen AWS IoT Greengrass ke perangkat.

Fitur ini tersedia untuk v2.3.0 dan versi kemudian dari komponen inti Greengrass.

Jenis sumber daya: Tidak ada

Format ARN sumber daya: *

`greengrass:ListThingGroupsForCoreDevice`

Memberikan izin untuk mendapatkan hierarki grup hal perangkat inti.

Izin ini diperiksa ketika perangkat inti menerima penerapan dariAWS IoT Greengrass. Perangkat inti menggunakan tindakan ini untuk mengidentifikasi apakah itu dihapus dari grup sesuatu sejak penerapan terakhir. Jika perangkat inti dihapus dari grup benda, dan grup benda itu adalah target penerapan ke perangkat inti, maka perangkat inti menghapus komponen yang diinstal oleh penerapan itu.

Fitur ini digunakan oleh v2.5.0 dan yang lebih baru dari komponen inti Greengrass.

Jenis sumber daya: `thing` (perangkat inti)

Format ARN sumber daya: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`core-device-thing-name`*

`greengrass:VerifyClientDeviceIdentity`

Memberikan izin untuk memverifikasi identitas perangkat klien yang tersambung ke perangkat inti.

Izin ini dievaluasi ketika perangkat inti menjalankan komponen autentikasi perangkat klien dan menerima koneksi MQTT dari perangkat klien. Perangkat klien menyajikan AWS IoT sertifikat perangkat. Kemudian, peranti inti mengirimkan sertifikat peranti ke AWS IoT Greengrass layanan cloud untuk memverifikasi identitas perangkat klien. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Jenis sumber daya: Tidak ada

Format ARN sumber daya: *

`greengrass:VerifyClientDeviceIoTCertificateAssociation`

Memberikan izin untuk memverifikasi apakah perangkat klien dikaitkan dengan AWS IoT sertifikat.

Izin ini dievaluasi ketika perangkat inti menjalankan komponen autentikasi perangkat klien dan mengotorisasi perangkat klien untuk terhubung melalui MQTT. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

> **Note**
>
> Agar perangkat inti bisa menggunakan operasi ini, Peran layanan Greengrass harus dikaitkan dengan Akun AWS dan memungkinkan izin `iot:DescribeCertificate`.

Jenis sumber daya: `thing` (perangkat klien)

Format ARN sumber daya: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`client-device-thing-name`*

`greengrass:PutCertificateAuthorities`

Memberikan izin untuk mengunggah sertifikat otoritas sertifikat (CA) bahwa perangkat klien dapat mengunduh untuk memverifikasi perangkat inti.

Izin ini dievaluasi ketika perangkat inti menginstal dan menjalankan komponen auth perangkat klien. Komponen ini menciptakan otoritas sertifikat lokal dan menggunakan operasi ini untuk mengunggah sertifikat CA. Perangkat klien mengunduh sertifikat CA ini ketika mereka menggunakan operasi Temukan untuk menemukan perangkat inti di mana mereka dapat terhubung. Ketika perangkat klien tersambung ke broker MQTT pada perangkat inti, mereka menggunakan sertifikat CA ini untuk memverifikasi identitas perangkat inti. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Jenis sumber daya: Tidak ada

Format ARN: *

`greengrass:GetConnectivityInfo`

Memberikan izin untuk mendapatkan informasi konektivitas untuk perangkat inti. Informasi ini menjelaskan bagaimana perangkat klien dapat terhubung ke perangkat inti.

Izin ini dievaluasi ketika perangkat inti menginstal dan menjalankan komponen auth perangkat klien. Komponen ini menggunakan informasi konektivitas untuk menghasilkan sertifikat CA yang valid untuk diunggah ke layanan AWS IoT Greengrass cloud dengan PutCertificateAuthoriesoperasi. Perangkat klien menggunakan sertifikat CA ini untuk memverifikasi identitas perangkat inti. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Anda juga dapat menggunakan operasi ini di bidang kendali AWS IoT Greengrass untuk melihat informasi konektivitas untuk perangkat inti. Untuk informasi selengkapnya, lihat GetConnectivityInfo di dalam Referensi API AWS IoT Greengrass V1.

Jenis sumber daya: `thing` (perangkat inti)

Format ARN sumber daya: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`core-device-thing-name`*

`greengrass:UpdateConnectivityInfo`

Memberikan izin untuk memperbarui informasi konektivitas untuk perangkat inti. Informasi ini menjelaskan bagaimana perangkat klien dapat terhubung ke perangkat inti.

Izin ini dievaluasi ketika perangkat inti menjalankan komponen detektor IP. Komponen ini mengidentifikasi informasi yang memerlukan perangkat klien untuk terhubung ke perangkat inti pada jaringan lokal. Kemudian, komponen ini menggunakan operasi ini untuk mengunggah informasi konektivitas ke layanan cloud AWS IoT Greengrass, sehingga perangkat klien dapat mengambil informasi ini dengan operasi Temukan. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Anda juga dapat menggunakan operasi ini di AWS IoT Greengrass bidang kontrol untuk secara manual memperbarui informasi konektivitas untuk perangkat inti. Untuk informasi selengkapnya, lihat UpdateConnectivityInfo di dalam Referensi API AWS IoT Greengrass V1.

Jenis sumber daya: `thing` (perangkat inti)

Format ARN sumber daya: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`core-device-thing-name`*

Tindakan perangkat klien

`greengrass:Discover`

Memberikan izin untuk menemukan informasi konektivitas untuk perangkat inti di mana perangkat klien dapat terhubung. Informasi ini menjelaskan bagaimana perangkat klien dapat terhubung ke perangkat inti. Perangkat klien hanya dapat menemukan perangkat inti yang telah Anda kaitkan dengan menggunakan BatchAssociateClientDeviceWithCoreDeviceoperasi. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

Jenis sumber daya: `thing` (perangkat klien)

Format ARN sumber daya: `arn:aws:iot:region:account-id:thing/client-device-thing-name`

Memperbarui AWS IoT kebijakan perangkat inti

Anda dapat menggunakan AWS IoT Greengrass dan AWS IoT konsol atau AWS IoT API untuk melihat dan memperbarui AWS IoT kebijakan perangkat inti.

> **Note**
>
> Jika Anda menggunakan Penginstal perangkat lunak inti AWS IoT Greengrass untuk penyediaan sumber daya, perangkat inti Anda memiliki kebijakan AWS IoT yang mengizinkan akses ke semua tindakan AWS IoT Greengrass (`greengrass:*`). Anda dapat mengikuti langkah-langkah ini untuk membatasi akses hanya ke tindakan yang digunakan oleh perangkat inti.

Meninjau dan memperbarui AWS IoT kebijakan perangkat inti (konsol)

1. Di menu navigasi konsol AWS IoT Greengrass tersebut, pilih Perangkat inti.
2. Pada halaman Perangkat inti, pilih perangkat inti yang akan diperbarui.
3. Pada halaman detail perangkat inti, pilih tautan ke Objek perangkat inti. Tautan ini membuka halaman rincian hal di AWS IoT konsol.
4. Pada halaman detail objek, pilih Sertifikat.
5. Di tab Sertifikat, pilih sertifikat aktif objek.
6. Pada halaman detail sertifikat, pilih Kebijakan.
7. Di tab Kebijakan, pilih kebijakan AWS IoT yang akan ditinjau dan diperbarui. Anda dapat menambahkan izin yang diperlukan untuk kebijakan yang dilampirkan ke sertifikat aktif perangkat inti.

   > **Note**
   >
   > Jika Anda menggunakan Penginstal perangkat lunak inti AWS IoT Greengrass untuk menyediakan sumber daya, Anda memiliki dua kebijakan AWS IoT. Kami menyarankan Anda memilih kebijakan yang diberi nama GreengrassV2IoTThingPolicy, jika ada. Perangkat inti yang Anda buat dengan penginstal cepat menggunakan nama kebijakan

ini secara default. Jika Anda menambahkan izin untuk kebijakan ini, Anda juga memberikan izin ini ke perangkat inti lain yang menggunakan kebijakan ini.

8. Dalam ikhtisar kebijakan, pilih Edit versi aktif.
9. Tinjau kebijakan dan tambahkan, hapus, atau edit izin sesuai kebutuhan.
10. Untuk menetapkan versi kebijakan baru sebagai versi aktif, di bawah Status versi Kebijakan, pilih Setel versi yang diedit sebagai versi aktif untuk kebijakan ini.
11. Pilih Simpan sebagai versi baru.

Meninjau dan memperbarui AWS IoT kebijakan perangkat inti (AWS CLI)

1. Buat daftar prinsipal untuk hal perangkat inti. AWS IoT Prinsipal hal dapat berupa sertifikat perangkat X.509 atau identifikasi lainnya. Jalankan perintah berikut, dan ganti *MyGreengrassCore*dengan nama perangkat inti.

   ```
   aws iot list-thing-principals --thing-name MyGreengrassCore
   ```

   Operasi mengembalikan respons yang mencantumkan prinsip hal perangkat inti.

   ```
   {
       "principals": [
           "arn:aws:iot:us-west-2:123456789012:cert/certificateId"
       ]
   }
   ```

2. Identifikasi sertifikat aktif perangkat inti. Jalankan perintah berikut, dan ganti *CertificateId* dengan ID setiap sertifikat dari langkah sebelumnya hingga Anda menemukan sertifikat aktif. ID sertifikat adalah string heksadesimal di akhir sertifikat ARN. `--query`Argumen menentukan untuk output hanya status sertifikat.

   ```
   aws iot describe-certificate --certificate-id certificateId --query
    'certificateDescription.status'
   ```

   Operasi mengembalikan status sertifikat sebagai string. Misalnya, jika sertifikat aktif, output `"ACTIVE"` operasi ini.

3. Buat daftar AWS IoT kebijakan yang dilampirkan pada sertifikat. Jalankan perintah berikut, dan ganti sertifikat ARN dengan ARN sertifikat.

```
aws iot list-principal-policies --principal arn:aws:iot:us-
west-2:123456789012:cert/certificateId
```

Operasi mengembalikan respons yang mencantumkan AWS IoT kebijakan yang dilampirkan pada sertifikat.

```
{
    "policies": [
        {
            "policyName":
 "GreengrassTESCertificatePolicyMyGreengrassCoreTokenExchangeRoleAlias",
            "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassTESCertificatePolicyMyGreengrassCoreTokenExchangeRoleAlias"
        },
        {
            "policyName": "GreengrassV2IoTThingPolicy",
            "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy"
        }
    ]
}
```

4. Pilih kebijakan yang akan dilihat dan diperbarui.

> **Note**
>
> Jika Anda menggunakan Penginstal perangkat lunak inti AWS IoT Greengrass untuk menyediakan sumber daya, Anda memiliki dua kebijakan AWS IoT. Kami menyarankan Anda memilih kebijakan yang diberi nama GreengrassV2IoTThingPolicy, jika ada. Perangkat inti yang Anda buat dengan penginstal cepat menggunakan nama kebijakan ini secara default. Jika Anda menambahkan izin untuk kebijakan ini, Anda juga memberikan izin ini ke perangkat inti lain yang menggunakan kebijakan ini.

5. Dapatkan dokumen kebijakan. Jalankan perintah berikut, dan ganti *GreenGrassV2IoT ThingPolicy* dengan nama kebijakan.

```
aws iot get-policy --policy-name GreengrassV2IoTThingPolicy
```

Operasi mengembalikan respons yang berisi dokumen kebijakan dan informasi lain tentang kebijakan tersebut. Dokumen kebijakan adalah objek JSON yang diserialisasikan sebagai string.

```
{
    "policyName": "GreengrassV2IoTThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
    "policyDocument": "{\
  \\"Version\\": \\"2012-10-17\\",\
  \\"Statement\\": [\
    {\
      \\"Effect\\": \\"Allow\\",\
      \\"Action\\": [\
                \\"iot:Connect\\",\
                \\"iot:Publish\\",\
                \\"iot:Subscribe\\",\
                \\"iot:Receive\\",\
                \\"greengrass:*\\"\
],\
      \\"Resource\\": \\"*\\"\
    }\
  ]\
}",
    "defaultVersionId": "1",
    "creationDate": "2021-02-05T16:03:14.098000-08:00",
    "lastModifiedDate": "2021-02-05T16:03:14.098000-08:00",
    "generationId":
 "f19144b798534f52c619d44f771a354f1b957dfa2b850625d9f1d0fde530e75f"
}
```

6. Gunakan konverter online atau alat lain untuk mengonversi string dokumen kebijakan menjadi objek JSON, lalu simpan ke file bernama`iot-policy.json`.

   Misalnya, jika Anda menginstal alat jq, Anda dapat menjalankan perintah berikut untuk mendapatkan dokumen kebijakan, mengubahnya menjadi objek JSON, dan menyimpan dokumen kebijakan sebagai objek JSON.

   ```
   aws iot get-policy --policy-name GreengrassV2IoTThingPolicy --query
    'policyDocument' | jq fromjson >> iot-policy.json
   ```

7. Tinjau dokumen kebijakan, dan tambahkan, hapus, atau edit izin sesuai kebutuhan.

Misalnya, pada sistem berbasis Linux, Anda dapat menjalankan perintah berikut untuk menggunakan GNU nano untuk membuka file.

```
nano iot-policy.json
```

Setelah selesai, dokumen kebijakan mungkin terlihat mirip dengan AWS IoTkebijakan minimal untuk perangkat inti.

8. Simpan perubahan sebagai versi baru kebijakan. Jalankan perintah berikut, dan ganti *GreenGrassV2IoT ThingPolicy* dengan nama kebijakan.

```
aws iot create-policy-version --policy-name GreengrassV2IoTThingPolicy --policy-
document file://iot-policy.json --set-as-default
```

Operasi mengembalikan respon mirip dengan contoh berikut jika berhasil.

```
{
    "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
    "policyDocument": "{\
  \\"Version\\": \\"2012-10-17\\",\
  \\"Statement\\": [\
    {\
      \\"Effect\\": \\"Allow\\",\
      \\"Action\\": [\
\\t\\t\\"iot:Connect\\",\
\\t\\t\\"iot:Publish\\",\
\\t\\t\\"iot:Subscribe\\",\
\\t\\t\\"iot:Receive\\",\
\\t\\t\\"greengrass:*\\"\
      ],\
      \\"Resource\\": \\"*\\"\
    }\
  ]\
}",
    "policyVersionId": "2",
    "isDefaultVersion": true
}
```

Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2

Important

Versi selanjutnya dari komponen inti Greengrass memerlukan izin tambahan pada kebijakan minimal. AWS IoT Anda mungkin perlu memperbarui AWS IoT kebijakan perangkat inti Anda untuk memberikan izin tambahan.

- Perangkat inti yang menjalankan Greengrass nucleus v2.5.0 dan yang lebih baru menggunakan `greengrass:ListThingGroupsForCoreDevice` izin untuk menghapus komponen saat Anda menghapus perangkat inti dari grup benda.
- Perangkat inti yang menjalankan Greengrass nucleus v2.3.0 dan yang lebih baru menggunakan `greengrass:GetDeploymentConfiguration` izin untuk mendukung dokumen konfigurasi penerapan besar.

Kebijakan contoh berikut mencakup serangkaian tindakan minimum yang diperlukan untuk mendukung fungsi Greengrass dasar untuk perangkat inti Anda.

- `connect`Kebijakan ini menyertakan * wildcard setelah nama perangkat inti (misalnya,*`core-device-thing-name`**). Perangkat inti menggunakan sertifikat perangkat yang sama untuk membuat beberapa langganan bersamaanAWS IoT Core, tetapi ID klien dalam koneksi mungkin tidak sama persis dengan nama perangkat inti. Setelah 50 langganan pertama, perangkat inti menggunakan *`core-device-thing-name`*#*`number`* sebagai ID klien, di mana *`number`* kenaikan untuk setiap tambahan 50 langganan. Misalnya, ketika perangkat inti bernama `MyCoreDevice` membuat 150 langganan bersamaan, ia menggunakan ID klien berikut:
  - Langganan 1 hingga 50: `MyCoreDevice`
  - Langganan 51 hingga 100: `MyCoreDevice#2`
  - Langganan 101 hingga 150: `MyCoreDevice#3`

  Wildcard memungkinkan perangkat inti untuk terhubung ketika menggunakan ID klien ini yang memiliki akhiran.
- Kebijakan ini mencantumkan topik MQTT dan filter topik yang perangkat inti dapat publikasikan, berlangganan, dan menerima pesan, termasuk topik yang digunakan untuk keadaan bayangan. Untuk mendukung pertukaran pesan antara AWS IoT Core, komponen Greengrass, dan perangkat klien, tentukan topik dan topik filter yang ingin Anda izinkan. Untuk informasi selengkapnya, lihat contoh kebijakan Publikasi/Berlangganan di Panduan Developer AWS IoT Core.

- Kebijakan tersebut memberikan izin untuk mempublikasikan topik berikut untuk data telemetri.

  ```
  $aws/things/core-device-thing-name/greengrass/health/json
  ```

  Anda dapat menghapus izin ini untuk perangkat inti di mana Anda menonaktifkan telemetri. Untuk informasi selengkapnya, lihat Kumpulkan data telemetri kondisi sistem dari perangkat inti AWS IoT Greengrass.

- Kebijakan ini memberikan izin untuk meneruskan IAM role melalui alias peran AWS IoT. Perangkat inti menggunakan peran ini, yang disebut peran pertukaran token, untuk memperoleh AWS kredensial yang dapat digunakan untuk mengautentikasi AWS permintaan. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

  Ketika Anda menginstal AWS IoT Greengrass Perangkat lunak inti, Anda membuat dan melampirkan kedua AWS IoT kebijakan yang hanya mencakup izin ini. Jika Anda menyertakan izin ini di kebijakan AWS IoT primer perangkat inti Anda, Anda dapat melepaskan dan menghapus kebijakan AWS IoT lainnya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:region:account-id:client/core-device-thing-name*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive",
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name/greengrass/health/json",
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name/greengrassv2/health/json",
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name/jobs/*",
```

```
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name/shadow/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-
thing-name/jobs/*",
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-
thing-name/shadow/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:AssumeRoleWithCertificate",
            "Resource": "arn:aws:iot:region:account-id:rolealias/token-exchange-role-
alias-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:GetComponentVersionArtifact",
                "greengrass:ResolveComponentCandidates",
                "greengrass:GetDeploymentConfiguration",
                "greengrass:ListThingGroupsForCoreDevice"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan AWS IoT minimal untuk mendukung perangkat klien

Kebijakan contoh berikut mencakup serangkaian tindakan minimum yang diperlukan untuk mendukung interaksi dengan perangkat klien pada perangkat inti. Untuk mendukung perangkat klien, perangkat inti harus memiliki izin pada kebijakan AWS IoT ini selain Kebijakan AWS IoT minimal untuk operasi dasar.

- Kebijakan ini memungkinkan perangkat inti untuk memperbarui informasi konektivitasnya sendiri. Izin (`greengrass:UpdateConnectivityInfo`) ini hanya diperlukan jika Anda menyebarkan komponen detektor IP ke perangkat inti.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name-gci/shadow/get"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-
thing-name-gci/shadow/update/delta",
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-
thing-name-gci/shadow/get/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name-gci/shadow/update/delta",
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name-gci/shadow/get/accepted"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "greengrass:PutCertificateAuthorities",
                "greengrass:VerifyClientDeviceIdentity"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:VerifyClientDeviceIoTCertificateAssociation"
            ],
            "Resource": "arn:aws:iot:region:account-id:thing/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:GetConnectivityInfo",
                "greengrass:UpdateConnectivityInfo"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/core-device-thing-name"
            ]
        }
    ]
}
```

Kebijakan AWS IoT minimal untuk perangkat klien

Kebijakan contoh berikut mencakup serangkaian tindakan minimum yang diperlukan untuk perangkat klien untuk menemukan perangkat inti di mana mereka terhubung dan berkomunikasi melalui MQTT. Kebijakan AWS IoT perangkat klien harus mencakup `greengrass:Discover` tindakan untuk memungkinkan perangkat menemukan informasi konektivitas untuk perangkat inti Greengrass yang terkait. Di bagian `Resource`, tentukan Amazon Resource Name (ARN) perangkat klien, bukan ARN perangkat inti Greengrass.

- Kebijakan ini memungkinkan komunikasi pada semua topik MQTT. Untuk mengikuti praktik keamanan terbaik, batasi `iot:Publish`, `iot:Subscribe`, dan `iot:Receive` ke set minimal perangkat klien yang tersambung ke perangkat inti untuk kasus penggunaan Anda.

- Kebijakan ini memungkinkan hal untuk menemukan perangkat inti untuk semua AWS IoT hal. Untuk mengikuti praktik keamanan terbaik, batasi izin `greengrass:Discover` ke objek AWS IoT perangkat klien atau wildcard yang cocok dengan satu rangkaian objek AWS IoT.

  > **Important**
  >
  > Variabel kebijakan objek (`iot:Connection.Thing.*`) tidak didukung untuk di kebijakan AWS IoT untuk perangkat inti atau operasi bidang data Greengrass. Sebaliknya, Anda dapat menggunakan wildcard yang cocok dengan beberapa perangkat yang memiliki nama yang sama. Misalnya, Anda dapat menentukan `MyGreengrassDevice*` untuk cocok dengan `MyGreengrassDevice1`, `MyGreengrassDevice2`, dan sebagainya.

- Kebijakan AWS IoT perangkat klien biasanya tidak memerlukan izin untuk tindakan `iot:GetThingShadow`, `iot:UpdateThingShadow`, atau `iot:DeleteThingShadow`, karena perangkat inti Greengrass tersebut menangani operasi sinkronisasi bayangan untuk perangkat klien. Untuk mengaktifkan perangkat inti untuk menangani bayangan perangkat klien, periksa bahwa AWS IoT kebijakan perangkat inti memungkinkan tindakan ini, dan bahwa bagian `Resource` termasuk ARN perangkat klien.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topicfilter/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:Discover"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/*"
            ]
        }
    ]
}
```

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	Ya
Kebijakan berbasis sumber daya	Tidak
Daftar kontrol akses (ACL)	Tidak
Otorisasi berbasis tag	Ya
Kredensi sementara	Ya
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 hal apa. Yaitu, principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam syarat apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan-tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya

memiliki nama yang sama sebagai operasi API AWS terkait. Ada beberapa pengecualian, misalnya tindakan hanya dengan izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Sertakan tindakan dalam kebijakan untuk memberikan izin guna melakukan operasi yang terkait.

Tindakan kebijakan di AWS IoT Greengrass menggunakan `greengrass:` awalan berikut sebelum tindakan. Misalnya, untuk memungkinkan seseorang untuk menggunakan Operasi API `ListCoreDevices` untuk mendaftar perangkat inti di Akun AWS, Anda menyertakan `greengrass:ListCoreDevices` tindakan dalam kebijakan mereka. Pernyataan kebijakan harus menyertakan elemen `Action` atau `NotAction`. AWS IoT Greengrass menentukan serangkaian tindakannya sendiri yang menjelaskan tugas yang dapat Anda 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 hal apa. Yaitu, principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam syarat apa.

Elemen kebijakan JSON `Resource` menentukan objek atau objek-objek yang menjadi target penerapan tindakan. Pernyataan harus mencakup elemen `Resource` atau `NotResource`. Sebagai praktik terbaik, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung tipe 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 tindakan AWS IoT Greengrass, lihat Tindakan yang Ditetapkan oleh AWS IoT Greengrass di Panduan Pengguna IAM.

Beberapa AWS IoT Greengrass tindakan (misalnya, 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 ARN, lihat Amazon Resource Name (ARN) dan namespaceAWS layanan dalam Referensi Umum Amazon Web Services.

Kunci syarat

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, prinsipal mana yang dapat melakukan tindakan pada sumber daya apa, dan menurut persyaratan apa.

Elemen `Condition` (atau `Condition` blok) memungkinkan Anda menentukan syarat di mana suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan operator syarat, seperti sama dengan atau kurang dari, untuk mencocokkan syarat dalam kebijakan dengan nilai dalam permintaan.

Jika Anda menentukan beberapa elemen `Condition` dalam pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya denga menggunakan operasi logika `AND`. Jika Anda menetapkan beberapa nilai untuk kunci syarat tunggal, AWS akan mengevaluasi syarat tersebut dengan menggunakan operasi logika `OR`. Semua persyaratan harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan syarat. Sebagai contoh, Anda dapat memberikan izin pengguna IAM untuk mengakses sumber daya hanya jika ditandai dengan nama pengguna IAM mereka. Untuk informasi lebih lanjut, lihat Elemen kebijakan IAM: variabel dan tag dalam Panduan Pengguna IAM.

AWS mendukung kunci syarat global dan kunci syarat khusus layanan. Untuk melihat semua kunci syarat global AWS, lihat Kunci konteks syarat global AWS dalam Panduan Pengguna IAM.

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 Beri tag pada sumber daya 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 kredensial keamanan sementara dengan memanggil operasiAWS STS API seperti AssumeRoleatau GetFederationToken.

Pada inti Greengrass, kredensial sementara untuk peran perangkat dibuat tersedia untuk komponen Greengrass. Jika komponen Anda menggunakan AWS SDK, Anda tidak perlu menambahkan logika untuk mendapatkan kredensial karena AWS SDK melakukan hal 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 perangkat inti menggunakan peran layanan untuk memungkinkan komponen Greengrass dan fungsi Lambda untuk mengakses beberapa AWS sumber daya atas nama Anda. Untuk informasi selengkapnya, lihat the section called “Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan”.

AWS IoT Greengrass menggunakan peran layanan untuk mengakses beberapa perangkat AWS sumber daya atas nama Anda. Untuk informasi selengkapnya, lihat Peran layanan Greengrass.

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 ini 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 IAM — 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 kebijakan untuk menetapkan 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.

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": "*"
        }
    ]
}
```

Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan

AWS IoT Greengrass perangkat inti menggunakan penyedia AWS IoT Core kredensi untuk mengotorisasi panggilan ke layanan. AWS Penyedia AWS IoT Core kredensi memungkinkan perangkat menggunakan sertifikat X.509 mereka sebagai identitas perangkat unik untuk mengautentikasi permintaan. AWS Ini menghilangkan kebutuhan untuk menyimpan ID kunci AWS akses dan kunci akses rahasia pada perangkat AWS IoT Greengrass inti Anda. Untuk informasi selengkapnya, lihat Mengotorisasi panggilan langsung ke AWS layanan di Panduan AWS IoT Core Pengembang.

Saat menjalankan perangkat lunak AWS IoT Greengrass Core, Anda dapat memilih untuk menyediakan AWS sumber daya yang dibutuhkan perangkat inti. Ini termasuk peran AWS Identity and Access Management (IAM) yang diasumsikan oleh perangkat inti Anda melalui penyedia AWS IoT Core kredensi. Gunakan `--provision true` argumen untuk mengonfigurasi peran dan kebijakan yang memungkinkan perangkat inti mendapatkan AWS kredensi sementara. Argumen ini juga mengonfigurasi alias AWS IoT peran yang menunjuk ke peran IAM ini. Anda dapat menentukan nama peran IAM dan alias AWS IoT peran yang akan digunakan. Jika Anda menentukan `--`

`provision true` tanpa parameter nama lain ini, perangkat inti Greengrass akan menciptakan dan menggunakan sumber daya default berikut:

- IAM role: `GreengrassV2TokenExchangeRole`

  Peran ini memiliki kebijakan bernama `GreengrassV2TokenExchangeRoleAccess` dan hubungan kepercayaan yang memungkinkan `credentials.iot.amazonaws.com` untuk menjalankan peran tersebut. Kebijakan ini mencakup izin minimum untuk perangkat inti.

  > **Important**
  >
  > Kebijakan ini tidak mencakup akses ke file dalam bucket S3. Anda harus menambahkan izin ke peran untuk mengizinkan perangkat inti mengambil artefak komponen dari bucket S3. Untuk informasi selengkapnya, lihat Izinkan akses ke bucket S3 untuk artefak komponen.

- AWS IoT alias peran: `GreengrassV2TokenExchangeRoleAlias`

  Alias peran ini mengacu pada IAM role.

Untuk informasi selengkapnya, lihat Langkah 3: Instal perangkat lunakAWS IoT Greengrass inti.

Anda juga dapat mengatur alias peran untuk perangkat inti yang ada. Untuk melakukannya, konfigurasikan parameter konfigurasi `iotRoleAlias` komponen inti Greengrass.

Anda dapat memperoleh AWS kredensi sementara untuk peran IAM ini untuk melakukan AWS operasi di komponen kustom Anda. Untuk informasi selengkapnya, lihat Berinteraksilah dengan layanan AWS.

Topik

- Izin peran layanan untuk perangkat inti
- Izinkan akses ke bucket S3 untuk artefak komponen

Izin peran layanan untuk perangkat inti

Peran memungkinkan layanan berikut untuk menjalankan peran tersebut:

- `credentials.iot.amazonaws.com`

Jika Anda menggunakan perangkat lunak AWS IoT Greengrass Inti untuk membuat peran ini, ia menggunakan kebijakan izin berikut untuk mengizinkan perangkat inti terhubung dan mengirim log ke AWS. Nama kebijakan default pada nama IAM role yang diakhiri dengan `Access`. Misalnya, jika Anda menggunakan nama IAM role default, maka nama kebijakan ini adalah `GreengrassV2TokenExchangeRoleAccess`.

Greengrass nucleus v2.5.0 and later

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

v2.4.x

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeCertificate",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
```

```
  ]
}
```

Earlier than v2.4.0

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeCertificate",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

Izinkan akses ke bucket S3 untuk artefak komponen

Peran perangkat inti default tidak mengizinkan perangkat inti mengakses bucket S3. Untuk men-deploy komponen yang memiliki artefak dalam bucket S3, Anda harus menambahkan izin `s3:GetObject` untuk mengizinkan perangkat inti mengunduh artefak komponen. Anda dapat menambahkan kebijakan baru ke peran perangkat inti untuk memberikan izin ini.

Untuk menambahkan kebijakan yang memungkinkan akses ke artefak komponen di Amazon S3

1. Buat file bernama `component-artifact-policy.json` dan salin JSON berikut ke dalam file. Kebijakan ini memungkinkan akses ke semua file dalam bucket S3. Ganti DOC-EXAMPLE-BUCKET dengan nama bucket S3 agar perangkat inti dapat mengakses.

   ```
   {
   ```

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

2. Jalankan perintah berikut untuk membuat kebijakan dari kebijakan dokumen di `component-artifact-policy.json`.

   Linux or Unix

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

   Windows Command Prompt (CMD)

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

   PowerShell

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

   Salin Amazon Resource Name (ARN) kebijakan dari metadata kebijakan dalam output. Anda menggunakan ARN ini untuk melampirkan kebijakan ini ke peran perangkat inti di langkah berikutnya.

3. Jalankan perintah berikut untuk melampirkan kebijakan tersebut pada peran perangkat inti. Ganti *GreenGrassV2 TokenExchangeRole* dengan nama peran yang Anda tentukan saat

menjalankan perangkat lunak Core. AWS IoT Greengrass Lalu, ganti ARN kebijakan dengan ARN dari langkah sebelumnya.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Jika perintah itu tidak memiliki output, ia berhasil, dan perangkat inti Anda dapat mengakses artefak yang Anda unggah ke bucket S3 ini.

Kebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya

Saat Anda menginstal perangkat lunak AWS IoT Greengrass Core, Anda dapat menyediakan AWS sumber daya yang diperlukan, seperti AWS IoT benda dan peran IAM untuk perangkat Anda. Anda juga dapat menyebarkan alat pengembangan lokal ke perangkat. Penginstal memerlukan AWS kredensi sehingga dapat melakukan tindakan ini di Anda. Akun AWS Untuk informasi selengkapnya, lihat Instal perangkat lunak inti AWS IoT Greengrass.

Kebijakan contoh berikut mencakup kumpulan tindakan minimum yang memerlukan installer untuk menyediakan sumber daya ini. Izin ini diperlukan jika Anda menentukan `--provision` argumen untuk penginstal. Ganti *account-id* dengan Akun AWS ID Anda, dan ganti *GreenGrassV2*

TokenExchangeRole dengan nama peran pertukaran token yang Anda tentukan dengan argumen installer. `--tes-role-name`

> **Note**
>
> Pernyataan kebijakan `DeployDevTools` diperlukan hanya jika Anda menentukan `--deploy-dev-tools` argumen untuk penginstal.

Greengrass nucleus v2.5.0 and later

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateTokenExchangeRole",
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreatePolicy",
                "iam:CreateRole",
                "iam:GetPolicy",
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::account-id:role/GreengrassV2TokenExchangeRole",
                "arn:aws:iam::account-
id:policy/GreengrassV2TokenExchangeRoleAccess",
                "arn:aws:iam::aws:policy/GreengrassV2TokenExchangeRoleAccess"
            ]
        },
        {
            "Sid": "CreateIoTResources",
            "Effect": "Allow",
            "Action": [
                "iot:AddThingToThingGroup",
                "iot:AttachPolicy",
                "iot:AttachThingPrincipal",
                "iot:CreateKeysAndCertificate",
                "iot:CreatePolicy",
                "iot:CreateRoleAlias",
```

```
                "iot:CreateThing",
                "iot:CreateThingGroup",
                "iot:DescribeEndpoint",
                "iot:DescribeRoleAlias",
                "iot:DescribeThingGroup",
                "iot:GetPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DeployDevTools",
            "Effect": "Allow",
            "Action": [
                "greengrass:CreateDeployment",
                "iot:CancelJob",
                "iot:CreateJob",
                "iot:DeleteThingShadow",
                "iot:DescribeJob",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:GetThingShadow",
                "iot:UpdateJob",
                "iot:UpdateThingShadow"
            ],
            "Resource": "*"
        }
    ]
}
```

Earlier than v2.5.0

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateTokenExchangeRole",
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreatePolicy",
                "iam:CreateRole",
                "iam:GetPolicy",
                "iam:GetRole",
```

```
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::account-id:role/GreengrassV2TokenExchangeRole",
                "arn:aws:iam::account-
id:policy/GreengrassV2TokenExchangeRoleAccess",
                "arn:aws:iam::aws:policy/GreengrassV2TokenExchangeRoleAccess"
            ]
        },
        {
            "Sid": "CreateIoTResources",
            "Effect": "Allow",
            "Action": [
                "iot:AddThingToThingGroup",
                "iot:AttachPolicy",
                "iot:AttachThingPrincipal",
                "iot:CreateKeysAndCertificate",
                "iot:CreatePolicy",
                "iot:CreateRoleAlias",
                "iot:CreateThing",
                "iot:CreateThingGroup",
                "iot:DescribeEndpoint",
                "iot:DescribeRoleAlias",
                "iot:DescribeThingGroup",
                "iot:GetPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DeployDevTools",
            "Effect": "Allow",
            "Action": [
                "greengrass:CreateDeployment",
                "iot:CancelJob",
                "iot:CreateJob",
                "iot:DeleteThingShadow",
                "iot:DescribeJob",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:GetThingShadow",
                "iot:UpdateJob",
                "iot:UpdateThingShadow"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

Peran layanan Greengrass

Peran layanan Greengrass adalah AWS Identity and Access Management peran layanan (IAM) yang mengotorisasi AWS IoT Greengrass untuk mengakses sumber daya dari AWS layanan atas nama Anda. Peran ini memungkinkan AWS IoT Greengrass untuk memverifikasi identitas perangkat klien dan mengelola informasi konektivitas perangkat inti.

> **Note**
>
> AWS IoT Greengrass V1 juga menggunakan peran ini untuk melakukan tugas-tugas penting. Untuk informasi selengkapnya, lihat Peran layanan Greengrass di Panduan Developer AWS IoT Greengrass V1.

Untuk mengizinkan AWS IoT Greengrass untuk mengakses sumber daya Anda, peran layanan Greengrass harus dikaitkan dengan Akun AWS dan tentukan AWS IoT Greengrasssebagai entitas terpercaya. Peran harus menyertakan kebijakan AWSGreengrassResourceAccessRolePolicyterkelola atau kebijakan khusus yang menentukan izin setara untuk AWS IoT Greengrass fitur yang Anda gunakan. AWSmempertahankan kebijakan ini, yang menentukan kumpulan izin yang AWS IoT Greengrass digunakan untuk mengakses sumber daya AndaAWS. Untuk informasi selengkapnya, lihat Kebijakan terkelola AWS: AWSGreengrassResourceAccessRolePolicy.

Anda dapat menggunakan kembali peran layanan Greengrass yang sama di Wilayah AWS, tetapi Anda harus mengaitkannya dengan akun Anda di setiap Wilayah AWS di mana Anda menggunakan AWS IoT Greengrass. Jika peran layanan tidak dikonfigurasi saat iniWilayah AWS, perangkat inti gagal memverifikasi perangkat klien dan gagal memperbarui informasi konektivitas.

Bagian berikut menjelaskan cara membuat dan mengelola peran layanan Greengrass dengan AWS Management Console atau AWS CLI.

Topik

- Kelola peran layanan Greengrass (konsol)
- Kelola peran layanan Greengrass (CLI)

- Lihat juga

> **Note**
>
> Selain peran layanan yang mengesahkan akses tingkat layanan, Anda menetapkan peran pertukaran token untuk perangkat inti Greengrass. Peran pertukaran token adalah IAM role terpisah yang mengendalikan bagaimana komponen Greengrass dan fungsi Lambda pada perangkat inti dapat mengakses layanan AWS. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Kelola peran layanan Greengrass (konsol)

Konsol AWS IoT membuatnya mudah untuk mengelola peran layanan Greengrass Anda. Sebagai contoh, ketika Anda mengonfigurasi penemuan perangkat klien untuk perangkat inti, konsol tersebut akan memeriksa apakah Akun AWS Anda melekat pada peran layanan Greengrass di Wilayah AWS saat ini. 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 konsol untuk tugas-tugas manajemen peran berikut:

Topik

- Temukan peran layanan Greengrass Anda (konsol)
- Buat peran layanan Greengrass (konsol)
- Ubah peran layanan Greengrass (konsol)
- Lapaskan peran layanan Greengrass (konsol)

> **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 gunakan di Wilayah AWS saat ini.

1. Navigasikan ke konsol AWS IoT tersebut.
2. Di panel navigasi, pilih Pengaturan.
3. Gulir ke Peran layanan Greengrass untuk melihat peran layanan dan kebijakannya.

   Jika Anda tidak melihat peran layanan, konsol dapat 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 Anda membuat peran ini dengan skrip penyiapan perangkat AWS IoT Greengrass V1, nama perannya adalah `GreengrassServiceRole_`*`random-string`*.

Ketika Anda mengonfigurasi penemuan perangkat klien untuk perangkat inti, konsol tersebut akan memeriksa apakah peran layanan Greengrass terkait dengan Akun AWS Anda di Wilayah AWS saat ini. Jika tidak, konsol akan meminta Anda untuk mengizinkan AWS IoT Greengrass untuk membaca dan menulis ke layanan AWS 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 menyarankan Anda juga menyertakan `aws:SourceArn` dan kunci konteks kondisi `aws:SourceAccount` global dalam kebijakan kepercayaan Anda untuk membantu mencegah masalah keamanan wakil yang membingungkan. Kunci konteks kondisi membatasi akses untuk mengizinkan hanya permintaan yang berasal dari akun tertentu dan ruang kerja Greengrass. Untuk informasi lebih lanjut tentang masalah wakil yang membingungkan, lihatCross-service bingung wakil pencegahan.
> Jika Anda membuat peran layanan, kembali ke konsol AWS IoT tersebut dan lampirkan peran tersebut ke Akun AWS Anda. Anda dapat melakukannya di bawah Peran layanan Greengrass pada halaman Pengaturan.

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. Navigasikan ke konsol AWS IoT tersebut.
2. Di panel navigasi, pilih Pengaturan.
3. 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.
4. Pilih peran layanan Greengrass untuk dilampirkan.
5. Pilih Lampirkan peran.

Lapaskan peran layanan Greengrass (konsol)

Gunakan prosedur berikut untuk melepaskan peran layanan Greengrass dari akun AWS di Wilayah AWS saat ini. Ini akan mencabut izin bagi AWS IoT Greengrass untuk mengakses layanan AWS di Wilayah AWSsaat ini.

> **Important**
>
> Melepaskan peran layanan dapat mengganggu operasi aktif.

1. Navigasikan ke AWS IoT konsol.
2. Di panel navigasi, pilih Pengaturan.
3. Di bawah Peran layanan Greengrass, pilih Lepaskan peran.
4. 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 memungkinkan AWS IoT Greengrass untuk mengambil izin atas nama Anda, di konsol IAM, pada halaman Peran, cari peran yang mencakup layanan AWS: Greengrass di kolom Entitas tepercaya.

Kelola peran layanan Greengrass (CLI)

Dalam prosedur berikut, kami berasumsi bahwa AWS Command Line Interface terinstal dan dikonfigurasi untuk menggunakan perangkat Akun AWS Anda. Untuk informasi selengkapnya, lihat Menginstal, memperbarui, dan melepas pemasangan AWS CLI dan Mengonfigurasi AWS CLI di Panduan Pengguna AWS Command Line Interface.

Anda dapat menggunakan AWS CLI untuk tugas-tugas manajemen peran berikut:

Topik

- Dapatkan peran layanan Greengrass (CLI)

- Buat peran layanan Greengrass (CLI)
- Hapus peran layanan Greengrass (CLI)

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 (misalnya, `us-west-2`).

  ```
  aws greengrassv2 get-service-role-for-account --region region
  ```

  Jika peran layanan Greengrass telah dikaitkan dengan akun Anda, permintaan akan mengembalikan metadata peran berikut.

  ```
  {
    "associatedAt": "timestamp",
    "roleArn": "arn:aws:iam::account-id:role/path/role-name"
  }
  ```

  Jika permintaan tidak mengembalikan metadata peran, maka Anda harus membuat peran layanan (jika tidak ada) dan mengaitkannya dengan akun Anda 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 mengambil peran. Contoh ini menciptakan peran bernama `Greengrass_ServiceRole`, tetapi Anda dapat menggunakan nama yang berbeda. Kami menyarankan Anda juga menyertakan `aws:SourceArn` dan kunci konteks kondisi `aws:SourceAccount` global dalam kebijakan kepercayaan Anda untuk membantu mencegah masalah keamanan wakil yang membingungkan. Kunci konteks kondisi membatasi akses untuk mengizinkan hanya permintaan yang berasal dari akun tertentu dan ruang kerja Greengrass. Untuk informasi lebih lanjut tentang masalah wakil yang membingungkan, lihatCross-service bingung wakil pencegahan.

Linux or Unix

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "greengrass.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:greengrass:region:account-id:*"
        },
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        }
      }
    }
  ]
}'
```

Windows Command Prompt (CMD)

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-
policy-document "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\":\"Allow\",\"Principal\":{\"Service\":\"greengrass.amazonaws.com\"},
\"Action\":\"sts:AssumeRole\",\"Condition\":{\"ArnLike\":{\"aws:SourceArn
\":\"arn:aws:greengrass:region:account-id:*\"},\"StringEquals\":
{\"aws:SourceAccount\":\"account-id\"}}}]}"
```

PowerShell

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "greengrass.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:greengrass:region:account-id:*"
        },
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        }
      }
    }
  ]
}'
```

2. 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 ARN peran layanan dan *`wilayah`* dengan Wilayah AWS (misalnya, `us-west-2`).

```
aws greengrassv2 associate-service-role-to-account --role-arn role-arn --
region region
```

Jika berhasil, permintaan tersebut akan mengembalikan tanggapan berikut.

```
{
  "associatedAt": "timestamp"
}
```

Hapus 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 greengrassv2 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-to-akun
  - disassociate-service-role-from-akun
  - get-service-role-for-akun
- Perintah IAM di Referensi PerintahAWS CLI
  - attach-role-policy

- menciptakan-peran
- hapus-peran
- delete-role-policy

Kebijakan terkelola AWS untuk AWS IoT Greengrass

SebuahAWSkebijakan terkelola adalah kebijakan mandiri yang dibuat dan dikelola olehAWS.AWSkebijakan terkelola dirancang untuk memberikan izin untuk banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwaAWSkebijakan terkelola mungkin tidak memberikan izin paling sedikit hak istimewa untuk kasus penggunaan spesifik Anda karena tersedia untuk semuaAWSpelanggan untuk digunakan. Kami menyarankan Anda mengurangi izin lebih lanjut dengan mendefinisikankebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalamAWSkebijakan yang dikelola. JikaAWSmemperbarui izin yang didefinisikan dalamAWSkebijakan terkelola, pembaruan mempengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan.AWSkemungkinan besar akan memperbaruiAWSkebijakan terkelola saat baruLayanan AWSdiluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat Kebijakan terkelola AWS dalam Panduan Pengguna IAM.

Topik

- Kebijakan terkelola AWS: AWSGreengrassFullAccess
- Kebijakan terkelola AWS: AWSGreengrassReadOnlyAccess
- Kebijakan terkelola AWS: AWSGreengrassResourceAccessRolePolicy
- AWS IoT Greengrass memperbarui pada kebijakan terkelola AWS

Kebijakan terkelola AWS: AWSGreengrassFullAccess

Anda dapat melampirkan kebijakan `AWSGreengrassFullAccess` ke identitas-identitas IAM Anda.

Kebijakan ini memberikan izin administratif yang memungkinkan akses penuh utama ke semua tindakan AWS IoT Greengrass.

Rincian perizinan

Kebijakan ini mencakup izin berikut:

- `greengrass` – Mengizinkan bagian utama untuk mendapatkan akses penuh ke semua tindakan AWS IoT Greengrass.

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

Kebijakan terkelola AWS: AWSGreengrassReadOnlyAccess

Anda dapat melampirkan kebijakan `AWSGreengrassReadOnlyAccess` ke identitas-identitas IAM Anda.

Kebijakan ini memberikan izin baca-saja yang memungkinkan prinsipal untuk melihat, tetapi tidak mengubah, informasi dalam AWS IoT Greengrass. Sebagai contoh, bagian utama dengan izin ini dapat melihat daftar komponen yang di-deploy ke perangkat inti Greengrass, tetapi tidak dapat membuat deployment untuk mengubah komponen yang berjalan pada perangkat tersebut.

Rincian perizinan

Kebijakan ini mencakup izin berikut:

- `greengrass` – Memungkinkan prinsipal untuk melakukan tindakan yang mengembalikan daftar item atau rincian tentang item. Ini termasuk operasi API yang dimulai dengan `List` atau `Get`.

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

```
            "Action": [
                "greengrass:List*",
                "greengrass:Get*"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan terkelola AWS: AWSGreengrassResourceAccessRolePolicy

Anda dapat melampirkan kebijakan `AWSGreengrassResourceAccessRolePolicy` pada entitas IAM Anda. AWS IoT Greengrass juga melampirkan kebijakan ini ke peran layanan yang memungkinkan AWS IoT Greengrass untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat Peran layanan Greengrass.

Kebijakan ini memberikan izin administratif yang memungkinkan AWS IoT Greengrassuntuk melakukan tugas-tugas penting, seperti mengambil fungsi Lambda Anda, mengelola bayangan perangkat AWS IoT, dan memverifikasi perangkat pelanggan Greengrass.

Rincian perizinan

Kebijakan ini mencakup izin berikut.

- `greengrass` — Kelola sumber daya Greengrass.
- `iot(*Shadow)` - MengelolaAWS IoTbayangan yang memiliki pengidentifikasi khusus berikut dalam nama mereka. Izin ini diperlukan sehingga AWS IoT Greengrass dapat berkomunikasi dengan perangkat inti.
  - `*-gci`—AWS IoT Greengrassmenggunakan bayangan ini untuk menyimpan informasi konektivitas perangkat inti, sehingga perangkat klien dapat menemukan dan terhubung ke perangkat inti.
  - `*-gcm`—AWS IoT Greengrass V1menggunakan bayangan ini untuk memberi tahu perangkat inti bahwa sertifikat otoritas sertifikat (CA) grup Greengrass telah diputar.
  - `*-gda`—AWS IoT Greengrass V1menggunakan bayangan ini untuk memberi tahu perangkat inti penyebaran.
  - `GG_*`— Tidak terpakai.
- `iot` (`DescribeThing` dan `DescribeCertificate`) — Ambil informasi tentang AWS IoT hal dan sertifikat. Izin ini diperlukan sehingga AWS IoT Greengrass dapat memverifikasi perangkat klien

yang tersambung ke perangkat inti. Untuk informasi selengkapnya, lihat Berinteraksilah dengan perangkat IoT lokal.

- `lambda` – Ambil informasi tentang fungsi AWS Lambda. Izin ini diperlukan sehingga AWS IoT Greengrass V1 dapat menyebarkan fungsi Lambda untuk inti Greengrass. Untuk informasi lebih lanjut, lihat Jalankan fungsi Lambda di inti AWS IoT Greengrass di Panduan Developer AWS IoT Greengrass V1.
- `secretsmanager` — Ambil nilai AWS Secrets Manager rahasia yang namanya dimulai dengan `greengrass-`. Izin ini diperlukan agar AWS IoT Greengrass V1 dapat men-deploy rahasia Secrets Manager pada inti Greengrass. Untuk informasi selengkapnya, lihat Sebarkan rahasia ke AWS IoT Greengrass inti di AWS IoT Greengrass V1 Panduan Pengembang.
- `s3` — Ambil objek file dari bucket S3 yang namanya berisi `greengrass` atau `sagemaker`. Izin ini diperlukan agar AWS IoT Greengrass V1 dapat men-deploy sumber daya machine learning yang Anda simpan di bucket S3. Untuk informasi selengkapnya, lihat sumber daya machine learning di Panduan Developer AWS IoT Greengrass V1.
- `sagemaker`— Ambil informasi tentang AmazonSageMakermodel inferensi pembelajaran mesin. Izin ini diperlukan agar AWS IoT Greengrass V1 dapat men-deploy model ML pada inti Greengrass. Untuk informasi selengkapnya, lihat Lakukan inferensi machine learning di Panduan Developer AWS IoT Greengrass V1.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowGreengrassAccessToShadows",
            "Action": [
                "iot:DeleteThingShadow",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iot:*:*:thing/GG_*",
                "arn:aws:iot:*:*:thing/*-gcm",
                "arn:aws:iot:*:*:thing/*-gda",
                "arn:aws:iot:*:*:thing/*-gci"
            ]
        },
        {
```

```
            "Sid": "AllowGreengrassToDescribeThings",
            "Action": [
                "iot:DescribeThing"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:*:*:thing/*"
        },
        {
            "Sid": "AllowGreengrassToDescribeCertificates",
            "Action": [
                "iot:DescribeCertificate"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:*:*:cert/*"
        },
        {
            "Sid": "AllowGreengrassToCallGreengrassServices",
            "Action": [
                "greengrass:*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Sid": "AllowGreengrassToGetLambdaFunctions",
            "Action": [
                "lambda:GetFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Sid": "AllowGreengrassToGetGreengrassSecrets",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:secretsmanager:*:*:secret:greengrass-*"
        },
        {
            "Sid": "AllowGreengrassAccessToS3Objects",
            "Action": [
                "s3:GetObject"
```

```
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::*Greengrass*",
                "arn:aws:s3:::*GreenGrass*",
                "arn:aws:s3:::*greengrass*",
                "arn:aws:s3:::*Sagemaker*",
                "arn:aws:s3:::*SageMaker*",
                "arn:aws:s3:::*sagemaker*"
            ]
        },
        {
            "Sid": "AllowGreengrassAccessToS3BucketLocation",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Sid": "AllowGreengrassAccessToSageMakerTrainingJobs",
            "Action": [
                "sagemaker:DescribeTrainingJob"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:sagemaker:*:*:training-job/*"
            ]
        }
    ]
}
```

AWS IoT Greengrass memperbarui pada kebijakan terkelola AWS

Anda dapat melihat detail tentang pembaruan kebijakan terkelola AWS untuk AWS IoT Greengrass sejak saat layanan mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganlah umpan RSS di halaman riwayat dokumen AWS IoT Greengrass V2.

Perubahan	Deskripsi	Tanggal
AWS IoT Greengrass mulai melacak perubahan	AWS IoT Greengrass mulai melacak perubahan untuk kebijakan terkelola AWS	2 Juli 2021

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 (yangdisebut 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::`*`account-id`*`:*`.

Untuk contoh kebijakan yang menggunakan`aws:SourceArn`dan`aws:SourceAccount`kunci konteks kondisi global, lihatBuat peran layanan Greengrass.

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
- Saya tidak memiliki izin untuk melakukan: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 Pemecahan 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 saat pengguna IAM `mateojackson` mencoba melihat detail tentang perangkat inti, tetapi tidak memiliki izin `greengrass:GetCoreDevice`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
 perform: greengrass:GetCoreDevice on resource: arn:aws:greengrass:us-
west-2:123456789012:coreDevices/MyGreengrassCore
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk memungkinkannya mengakses sumber daya `arn:aws:greengrass:us-west-2:123456789012:coreDevices/MyGreengrassCore` dengan menggunakan tindakan `greengrass:GetCoreDevice`.

Berikut adalah masalah IAM umum yang mungkin Anda hadapi saat bekerja dengan AWS IoT Greengrass.

Saya tidak memiliki izin untuk melakukan:PassRole

Jika Anda menerima pesan kesalahan bahwa Anda tidak terotorisasi untuk melakukan`iam:PassRole`tindakan, kebijakan Anda harus diperbarui untuk mengizinkan Anda dapat memberikan peran keAWS IoT Greengrass.

BeberapaLayanan AWSmengizinkan Anda dapat meneruskan peran yang sudah ada ke layanan tersebut alih-alih membuat peran layanan atau peran tertaut layanan baru. 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 agar dia mendapatkan izin`iam:PassRole`tindakan.

Jika Anda membutuhkan bantuan, hubungiAWSadministrator. Administrator Anda adalah orang yang memberikan kredensi masuk Anda.

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 selengkapnya, lihat Menyediakan akses ke pengguna IAM di Akun AWS lainnya yang Anda miliki dan Menyediakan akses ke Akun AWS 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).

Izinkan lalu lintas perangkat melalui proxy atau firewall

Perangkat inti Greengrass dan komponen Greengrass melakukan permintaan keluar ke layanan dan situs web lain. AWS Sebagai langkah keamanan, Anda dapat membatasi lalu lintas keluar ke sejumlah kecil titik akhir dan port. Anda dapat menggunakan informasi berikut tentang titik akhir dan port untuk membatasi lalu lintas perangkat melalui proxy, firewall, atau grup keamanan Amazon VPC. Untuk informasi selengkapnya tentang cara mengonfigurasi perangkat inti agar menggunakan proxy, lihatHubungkan pada port 443 atau melalui proksi jaringan.

Topik

- Titik akhir untuk operasi dasar
- Titik akhir untuk instalasi dengan penyediaan otomatis
- Titik akhir untuk komponen AWS yang disediakan

Titik akhir untuk operasi dasar

Perangkat inti Greengrass menggunakan titik akhir dan port berikut untuk operasi dasar.

Ambil titik akhir AWS IoT

Dapatkan titik akhir AWS IoT untuk Akun AWS, dan simpan untuk digunakan nanti. Perangkat Anda menggunakan titik akhir ini untuk tersambung ke AWS IoT. Lakukan hal-hal berikut:

1. Dapatkan titik akhir data AWS IoT untuk perangkat Akun AWS Anda.

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

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
   ```

```
}
```

2. Dapatkan titik akhir kredensial AWS IoT untuk Akun AWS Anda.

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

Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

```
{
  "endpointAddress": "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
}
```

Titik Akhir	Port	Diperlukan	Deskripsi
`greengrass-ats.iot. region.amazonaws.com`	8443 atau 443	Ya	Digunakan untuk operasi pesawat data, seperti menginstal penerapan dan bekerja dengan perangkat klien.
`device-data-prefix -ats.iot. region.amazonaws.com`	MQTT: 8883 atau 443 HTTPS: 8443 atau 443	Ya	Digunakan untuk operasi bidang data untuk manajemen perangkat, seperti

Titik Akhir	Port	Diperlukan	Deskripsi
			komunikasi MQTT dan sinkronisasi bayangan dengan AWS IoT Core.

Titik Akhir	Port	Diperlukan	Deskripsi
device-credentials-prefix .credentials.iot. *region*.amazonaws.com	443	Ya	Digunakan untuk memperoleh kredensial AWS, yang digunakan oleh perangkat inti untuk mengunduh artefak komponen dari Amazon S3 dan melakukan operasi lainnya. Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

Titik Akhir	Port	Diperlukan	Deskripsi

`*.s3.amazonaws.com`
 	
`*.s3.region.amazonaws.com`
	443	Ya	Digunakan untuk deployment. Format ini mencakup karakter *, karena prefiks titik akhir dikendalikan secara internal dan mungkin berubah setiap saat.

Titik Akhir	Port	Diperlukan	Deskripsi
`data.iot.` *`region`*`.amazonaws.com`	443	Tidak	Diperlukan jika perangkat inti menjalankan versi inti Greengrass lebih awal dari v2.4.0 dan dikonfigurasi untuk menggunakan proxy jaringan. Perangkat inti menggunakan titik akhir ini untuk komunikasi MQTT dengan AWS IoT Core ketika berada di belakang proksi. Untuk informasi selengkapnya, lihat Konfigura

Titik Akhir	Port	Diperlukan	Deskripsi
			sikan proksi jaringan.

Titik akhir untuk instalasi dengan penyediaan otomatis

Perangkat inti Greengrass menggunakan titik akhir dan port berikut saat Anda menginstal AWS IoT Greengrass perangkat lunak Core dengan penyediaan sumber daya otomatis.

Titik Akhir	Port	Diperlukan	Deskripsi
`iot.region.amazonaws.com`	443	Ya	Digunakan untuk membuat AWS IoT sumber daya dan mengambil informasi tentang AWS IoT sumber daya yang ada.
`iam.amazonaws.com`	443	Ya	Digunakan untuk membuat sumber daya IAM dan mengambil informasi tentang sumber

Titik Akhir	Port	Diperlukan	Deskripsi
			daya IAM yang ada.
`sts.region.amazonaws.com`	443	Ya	Digunakan untuk mendapatkan ID AndaAkun AWS.
`greengrass. region.amazonaws.com`	443	Tidak	Diperlukan jika Anda menggunakan `--deploy-dev-tools` argumen untuk menyebarkan komponen CLI Greengrass ke perangkat inti.

Titik akhir untuk komponen AWS yang disediakan

Perangkat inti Greengrass menggunakan titik akhir tambahan tergantung pada komponen perangkat lunak mana yang mereka jalankan. Anda dapat menemukan titik akhir yang dibutuhkan oleh setiap komponen AWS yang disediakan di bagian Persyaratan di setiap halaman komponen dalam panduan pengembang ini. Lihat informasi yang lebih lengkap di Komponen yang disediakan oleh AWS.

Validasi kepatuhan untuk AWS IoT Greengrass

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program 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 oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar AWS yang berfokus pada keamanan dan kepatuhan.
- Arsitektur untuk Keamanan dan Kepatuhan HIPAA di Amazon Web Services — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.

  > **Note**
  >
  > Tidak semua memenuhi Layanan AWS syarat HIPAA. Untuk informasi selengkapnya, lihat Referensi Layanan yang Memenuhi Syarat HIPAA.

- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Panduan Kepatuhan Pelanggan - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber

daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.

- Amazon GuardDuty — Ini Layanan AWS mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCI DSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- AWS Audit ManagerIni Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan 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. Availability Zone lebih tersedia, toleran kegagalan, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau banyak yang tradisional.

Untuk informasi lebih lanjut, lihat AWS Infrastruktur Global.

Selain infrastruktur global AWS, AWS IoT Greengrass menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan backup Anda.

- Anda dapat mengonfigurasi perangkat inti Greengrass untuk menulis catatan ke sistem file lokal dan CloudWatch Log. Jika perangkat core kehilangan konektivitas, perangkat dapat terus mencatat pesan pada sistem file. Ketika menghubungkan kembali, ia menulis pesan log untuk CloudWatch Log. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.
- Jika perangkat inti kehilangan daya selama penyebaran, perangkat tersebut akan melanjutkan penerapan setelahAWS IoT GreengrassPerangkat lunak inti dimulai lagi.
- Jika perangkat core kehilangan konektivitas internet, perangkat klien Greengrass dapat terus berkomunikasi melalui jaringan lokal.

- Anda dapat menulis komponen Greengrass yang membaca pengaliran manajer pengaliran dan mengirim data ke tujuan penyimpanan lokal.

Keamanan infrastruktur dalam AWS IoT Greengrass

Sebagai suatu layanan terkelola, AWS IoT Greengrass dilindungi oleh prosedur keamanan jaringan global AWS yang dijelaskan dalam laporan resmi Amazon Web Services: Gambaran Umum Proses Keamanan.

Anda menggunakan panggilan API yang dipublikasikan AWS untuk mengakses AWS IoT Greengrass melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.2 atau versi yang lebih baru. Kami merekomendasikan TLS 1.3 atau versi yang lebih baru. Selain itu, klien harus mendukung cipher suites dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Sebagian besar sistem-sistem modern seperti Java 7 dan versi yang lebih baru mendukung mode-mode ini.

Permintaan harus ditandatangani menggunakan access key ID dan secret access key yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara untuk 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 memungkinkan untuk menerapkan kebijakan keamanan yang konsisten di seluruh perangkat Anda dan merespons dengan cepat saat perangkat disusupi. Untuk informasi selengkapnya, lihat topik berikut:

- Komponen Pertahanan Perangkat
- AWS IoT Device Defender di Panduan Developer AWS IoT Core.

Di lingkungan AWS IoT Greengrass, Anda harus mengetahui pertimbangan berikut ini:

- 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 Greengrass yang ditetapkan pengguna, apakah mereka berjalan dalam kontainer Greengrass atau tidak. Oleh karena itu, mungkin bagi komponen Greengrass untuk berkomunikasi dengan proses lain yang berjalan di sistem atau di luar melalui jaringan.

Integritas kode diAWS IoT Greengrass V2

AWS IoT Greengrassmenyebarkan komponen perangkat lunak dariAWS Cloudke perangkat yang menjalankanAWS IoT GreengrassPerangkat lunak inti. Komponen perangkat lunak ini termasukAWSKomponen yang disediakandankomponen kustomyang Anda unggah keAkun AWS. Setiap komponen terdiri atas resep. Resep mendefinisikan metadata komponen, dan sejumlah artefak, yang merupakan biner komponen, seperti kode dikompilasi dan sumber daya statis. Artefak komponen disimpan di Amazon S3.

Saat Anda mengembangkan dan menyebarkan komponen Greengrass, Anda mengikuti langkah-langkah dasar ini yang bekerja dengan artefak komponen diAkun AWSdan pada perangkat Anda:

1. Buat dan unggah artefak ke bucket S3.
2. Buat komponen dari resep dan artefak diAWS IoT Greengrasslayanan, yang menghitunghash kriptografidari setiap artefak.
3. Menyebarkan komponen ke perangkat inti Greengrass, yang men-download dan memverifikasi integritas masing-masing artefak.

AWSbertanggung jawab untuk menjaga integritas artefak setelah Anda mengunggah artefak ke ember S3, termasuk saat Anda menyebarkan komponen ke perangkat inti Greengrass. Anda bertanggung jawab untuk mengamankan artefak perangkat lunak sebelum mengunggah artefak ke ember S3. Anda juga bertanggung jawab untuk mengamankan akses ke sumber daya diAkun AWS, termasuk ember S3 tempat Anda mengunggah artefak komponen.

> **Note**
>
> Amazon S3 menyediakan fitur bernama S3 Object Lock yang dapat Anda gunakan untuk melindungi terhadap perubahan artefak komponen dalam ember S3Akun AWS. Anda dapat menggunakan S3 Object Lock untuk mencegah artefak komponen terhapus atau ditimpa. Untuk informasi selengkapnya, lihatMenggunakan S3 Object LockdiPanduan Pengguna Amazon Simple Storage.

SaatAWSmenerbitkan komponen publik, dan ketika Anda mengunggah komponen kustom,AWS IoT Greengrassmenghitung digest kriptografi untuk setiap artefak komponen.AWS IoT Greengrassmemperbarui resep komponen untuk memasukkan setiap artefak digest dan algoritma hash yang digunakan untuk menghitung mencerna itu. Pencernaan ini menjamin integritas artefak, karena jika artefak berubahAWS Cloudatau selama download, file-nya digest tidak akan cocok dengan digest yangAWS IoT Greengrasstoko dalam resep komponen. Untuk informasi selengkapnya, lihatArtefak dalam referensi resep komponen.

Ketika Anda men-deploy komponen ke perangkat inti,AWS IoT GreengrassPerangkat lunak inti mendownload resep komponen dan setiap artefak komponen yang ditentukan oleh resep. ParameterAWS IoT GreengrassPerangkat lunak inti menghitung digest dari setiap file artefak download dan membandingkannya dengan yang artefak digest dalam resep. Jika digests tidak cocok, deployment gagal, danAWS IoT GreengrassPerangkat lunak inti menghapus artefak yang diunduh dari sistem file perangkat. Untuk informasi selengkapnya tentang cara koneksi antara perangkat inti danAWS IoT Greengrassdiamankan, lihatEnkripsi dalam transit.

Anda bertanggung jawab untuk mengamankan file artefak komponen pada sistem file perangkat inti Anda. ParameterAWS IoT GreengrassPerangkat lunak inti menyimpan artefak kepackagesfolder dalam folder akar Greengrass. Anda dapat menggunakanAWS IoT Device Defenderuntuk menganalisis, mengaudit, dan memonitor perangkat inti. Untuk informasi selengkapnya, lihat Analisis konfigurasi dan kerentanan dalam AWS IoT Greengrass.

AWS IoT Greengrass dan titik akhir VPC antarmuka (AWS PrivateLink)

Anda dapat membuat koneksi pribadi antara VPC Anda dan bidang AWS IoT Greengrass kontrol dengan membuat titik akhir VPC antarmuka. Anda dapat menggunakan endpoint ini untuk mengelola komponen, penerapan, dan perangkat inti dalam layanan. AWS IoT Greengrass Endpoint antarmuka didukung oleh AWS PrivateLink, teknologi yang memungkinkan Anda mengakses AWS IoT Greengrass API secara pribadi tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi Direct AWS 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.

Setiap titik akhir antarmuka diwakili oleh satu atau lebih Antarmuka Jaringan Elastis dalam 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 titik akhir VPC antarmuka untuk AWS IoT Greengrass operasi bidang kontrol
- Membuat kebijakan VPC endpoint untuk AWS IoT Greengrass
- Mengoperasikan perangkat AWS IoT Greengrass inti di VPC

Pertimbangan untuk VPC endpoint AWS IoT Greengrass

Sebelum menyiapkan titik akhir VPC antarmukaAWS IoT Greengrass, tinjau properti dan batasan titik akhir Antarmuka di Panduan Pengguna Amazon VPC. Selain itu, perhatikan pertimbangan berikut:

- AWS IoT Greengrassmendukung panggilan ke semua tindakan API bidang kontrolnya dari VPC Anda. Pesawat kontrol mencakup operasi seperti CreateDeploymentdan ListEffectiveDeployments. Pesawat kontrol tidak termasuk operasi seperti ResolveComponentCandidatesdan Discover, yang merupakan operasi pesawat data.
- Titik akhir VPC untuk AWS IoT Greengrass saat ini tidak didukung di AWS Wilayah China.

Buat titik akhir VPC antarmuka untuk AWS IoT Greengrass operasi bidang kontrol

Anda dapat membuat titik akhir VPC untuk bidang AWS IoT Greengrass kontrol menggunakan konsol VPC Amazon atau (). AWS Command Line Interface AWS CLI Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

Buat titik akhir VPC untuk AWS IoT Greengrass menggunakan nama layanan berikut:

- com.amazonaws. *wilayah* .greengrass

Jika Anda mengaktifkan DNS pribadi untuk titik akhir, Anda dapat membuat permintaan API untuk AWS IoT Greengrass menggunakan nama DNS default untuk Wilayah, misalnya,. `greengrass.us-east-1.amazonaws.com` DNS pribadi 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 titik akhir VPC yang mengontrol akses AWS IoT Greengrass untuk mengontrol operasi pesawat. Kebijakan titik akhir menentukan informasi berikut:

- Prinsip yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan oleh prinsip.
- Sumber daya yang dapat dilakukan oleh kepala sekolah.

Untuk informasi lebih lanjut, lihat Mengendalikan akses ke layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Example Contoh: Kebijakan VPC endpoint untuk tindakan AWS IoT Greengrass

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:ListEffectiveDeployments"
            ],
            "Resource": "*"
        }
    ]
}
```

Mengoperasikan perangkat AWS IoT Greengrass inti di VPC

Anda dapat mengoperasikan perangkat inti Greengrass dan melakukan penerapan di VPC tanpa akses internet publik. Minimal, Anda harus mengatur titik akhir VPC berikut dengan alias DNS yang sesuai. Untuk informasi selengkapnya tentang cara membuat dan menggunakan titik akhir VPC, lihat Membuat titik akhir VPC di Panduan Pengguna Amazon VPC.

> **Note**
>
> Fitur VPC untuk membuat catatan DNS secara otomatis dinonaktifkan untuk AWS IoT data dan Kredensialnya. AWS IoT Untuk menghubungkan titik akhir ini, Anda harus membuat catatan DNS Pribadi secara manual. Untuk informasi selengkapnya, lihat DNS pribadi untuk titik akhir antarmuka. Untuk informasi selengkapnya tentang batasan AWS IoT Core VPC, lihat Batasan titik akhir VPC.

Prasyarat

- Anda harus menginstal perangkat lunak AWS IoT Greengrass Core menggunakan langkah-langkah penyediaan manual. Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya manual.

Batasan

- Mengoperasikan perangkat inti Greengrass di VPC tidak didukung di Wilayah China dan. AWS GovCloud (US) Regions
- Untuk informasi selengkapnya tentang batasan AWS IoT data dan titik akhir VPC penyedia AWS IoT kredensi, lihat Batasan.

Siapkan perangkat inti Greengrass Anda untuk beroperasi di VPC

1. Dapatkan titik akhir AWS IoT untuk Akun AWS, dan simpan untuk digunakan nanti. Perangkat Anda menggunakan titik akhir ini untuk tersambung ke AWS IoT. Lakukan hal-hal berikut:

   a. Dapatkan titik akhir data AWS IoT untuk perangkat Akun AWS Anda.

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

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
   }
   ```

   b. Dapatkan titik akhir kredensial AWS IoT untuk Akun AWS Anda.

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

   Respons tersebut serupa dengan contoh berikut ini, jika permintaannya berhasil.

   ```
   {
     "endpointAddress": "device-credentials-prefix.credentials.iot.us-
   west-2.amazonaws.com"
   }
   ```

2. Buat antarmuka Amazon VPC untuk AWS IoT data dan titik akhir AWS IoT kredensialnya:

   a. Arahkan ke konsol VPC Endpoints, di bawah Virtual private cloud di menu sebelah kiri, pilih Endpoints lalu Create Endpoint.

   b. Di halaman Buat titik akhir, tentukan informasi berikut.

   - Pilih Layanan AWSs untuk kategori Layanan.
   - Untuk Nama Layanan, cari dengan memasukkan kata kunci`iot`. Dalam daftar `iot` layanan yang ditampilkan, pilih titik akhir.

     Jika Anda membuat titik akhir VPC untuk bidang AWS IoT Core data, pilih titik akhir API bidang AWS IoT Core data untuk Wilayah Anda. Titik akhir akan menjadi format`com.amazonaws.region.iot.data`.

Jika Anda membuat titik akhir VPC untuk penyedia AWS IoT Core kredensi, pilih titik akhir penyedia AWS IoT Core kredensi untuk Wilayah Anda. Titik akhir akan menjadi formatcom.amazonaws.*region*.iot.credentials.

> **Note**
>
> Nama layanan untuk pesawat AWS IoT Core data di Wilayah China akan menjadi formatcn.com.amazonaws.*region*.iot.data. Membuat titik akhir VPC untuk penyedia AWS IoT Core kredensi tidak didukung di Wilayah China.

- Untuk VPC dan Subnet, pilih VPC tempat Anda ingin membuat titik akhir, dan Availability Zones (AZ) tempat Anda ingin membuat jaringan endpoint.
- Untuk Aktifkan nama DNS, pastikan Aktifkan untuk titik akhir ini tidak dipilih. Baik pesawat AWS IoT Core data maupun penyedia AWS IoT Core kredensi belum mendukung nama DNS pribadi.
- Untuk grup Keamanan, pilih grup keamanan yang ingin Anda kaitkan dengan antarmuka jaringan titik akhir.
- Secara opsional, Anda dapat menambah atau menghapus tag. Tag adalah pasangan nama-nilai yang Anda gunakan untuk mengaitkan dengan titik akhir Anda.

c. Untuk membuat titik akhir VPC Anda, pilih Buat titik akhir.

3. Setelah Anda membuat AWS PrivateLink titik akhir, di tab Detail titik akhir Anda, Anda akan melihat daftar nama DNS. Anda dapat menggunakan salah satu nama DNS yang Anda buat di bagian ini untuk mengonfigurasi zona host pribadi Anda.
4. Buat titik akhir Amazon S3. Untuk informasi selengkapnya, lihat Membuat titik akhir VPC untuk Amazon S3.
5. Jika Anda menggunakan komponen Greengrass AWS -provided, titik akhir dan konfigurasi tambahan mungkin diperlukan. Untuk melihat persyaratan titik akhir, pilih komponen dari daftar komponen AWS yang disediakan dan lihat bagian Persyaratan. Misalnya, persyaratan komponen pengelola log menyarankan bahwa komponen ini harus dapat melakukan permintaan keluar ke titik akhirlogs.*region*.amazonaws.com.

   Jika Anda menggunakan komponen Anda sendiri, Anda mungkin perlu meninjau dependensi dan melakukan pengujian tambahan untuk menentukan apakah ada titik akhir tambahan yang diperlukan.

6. Dalam konfigurasi `greengrassDataPlaneEndpoint` inti Greengrass, harus diatur ke. **`iotdata`** Untuk informasi selengkapnya, lihat konfigurasi nukleus Greengrass.
7. Jika Anda berada di `us-east-1` wilayah tersebut, atur parameter konfigurasi `s3EndpointType` ke **`REGIONAL`** dalam konfigurasi inti Greengrass. Fitur ini tersedia untuk Greengrass nucleus versi 2.11.3 atau yang lebih baru.

Example Contoh: Konfigurasi komponen

```
{
"aws.greengrass.Nucleus": {
   "configuration": {
      "awsRegion": "us-east-1",
      "iotCredEndpoint": "xxxxxx.credentials.iot.region.amazonaws.com",
      "iotDataEndpoint": "xxxxxx-ats.iot.region.amazonaws.com",
      "greengrassDataPlaneEndpoint": "iotdata",
      "s3EndpointType": "REGIONAL"
      ...
     }
   }
}
```

Tabel berikut memberikan informasi tentang alias DNS pribadi kustom yang sesuai.

Layanan	Nama layanan titik akhir VPC	Jenis titik akhir VPC	Alias DNS pribadi khusus	Catatan
AWS IoT data	`com.amazonaws.` *`region`*`.iot.d`	Antarmuka	*`prefix`*`-ats.iot.`*`region`*`.a` `s.com`	Catatan DNS pribadi harus sesuai dengan AWS IoT data titik akhir akun

Layanan	Nama layanan titik akhir VPC	Jenis titik akhir VPC	Alias DNS pribadi khusus	Catatan
				Anda:. `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
Kredesial AWS IoT	`com.amazonaws.`*`region`*`.iot.credentials`	Antarmuka	*`prefix`*`.credentials.iot.`*`region`*`.amazonaws.com`	Catatan DNS pribadi harus sesuai dengan titik akhir AWS IoT Kredensial akun Anda:. `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`

Layanan	Nama layanan titik akhir VPC	Jenis titik akhir VPC	Alias DNS pribadi khusus	Catatan
Amazon S3	`com.amazonaws.` *region*`.s3`	Antarmuka		Catatan DNS dibuat secara otomatis.

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 untuk komponen Anda dengan menjalankannya sebagai pengguna yang tidak memiliki hak istimewa. Komponen tidak boleh berjalan sebagai root kecuali benar-benar diperlukan.

Gunakan set izin minimum dalam peran IAM. 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 hak istimewa setidaknya juga berlaku untuk kebijakan AWS IoT yang Anda lampirkan ke inti Greengrass Anda.

Jangan kode keras kredensial dalam komponen Greengrass

Jangan membuat kode keras pada kredensial dalam komponen Greengrass yang ditentukan pengguna Anda. Untuk melindungi kredensial Anda dengan lebih baik:

- Untuk berinteraksi dengan layanan AWS, tentukan izin untuk tindakan tertentu dan sumber daya dalam Peran layanan perangkat inti Greengrass.
- Gunakan komponen secret manager untuk menyimpan kredensial Anda. Atau, jika fungsi tersebut menggunakan SDK AWS, 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 membutuhkan izin IAM.

- Jangan gunakan informasi sensitif di jalur topik MQTT.
- Jangan gunakan informasi sensitif pada nama, jenis, dan atribut perangkat (objek) di registri AWS IoT Core.
- Jangan log informasi sensitif dalam komponen Greengrass yang ditetapkan pengguna Anda atau fungsi Lambda.
- Jangan gunakan informasi sensitif dalam nama dan ID sumber daya Greengrass:
  - Perangkat inti
  - Komponen
  - Deployment
  - Pencatat

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 selengkapnya, lihat praktik terbaik Terus sinkronkan jam perangkat di Panduan Developer AWS IoT Core.

Rekomendasi Cipher Suite

Greengrass default memilih TLS Cipher Suites terbaru yang tersedia di perangkat. Pertimbangkan untuk menonaktifkan penggunaan suite cipher lama di perangkat. Misalnya, suite cipher CBC.

Untuk informasi lebih lanjut, lihatKonfigurasi Kriptografi Java.

Lihat juga

- Praktik terbaik keamanan di AWS IoT Core pada Panduan Developer AWS IoT
- Sepuluh aturan emas keamanan untuk solusi IoT IndustripadaInternet of Things diAWSBlog Resmi

Menggunakan AWS IoT Device Tester untuk AWS IoT Greengrass V2

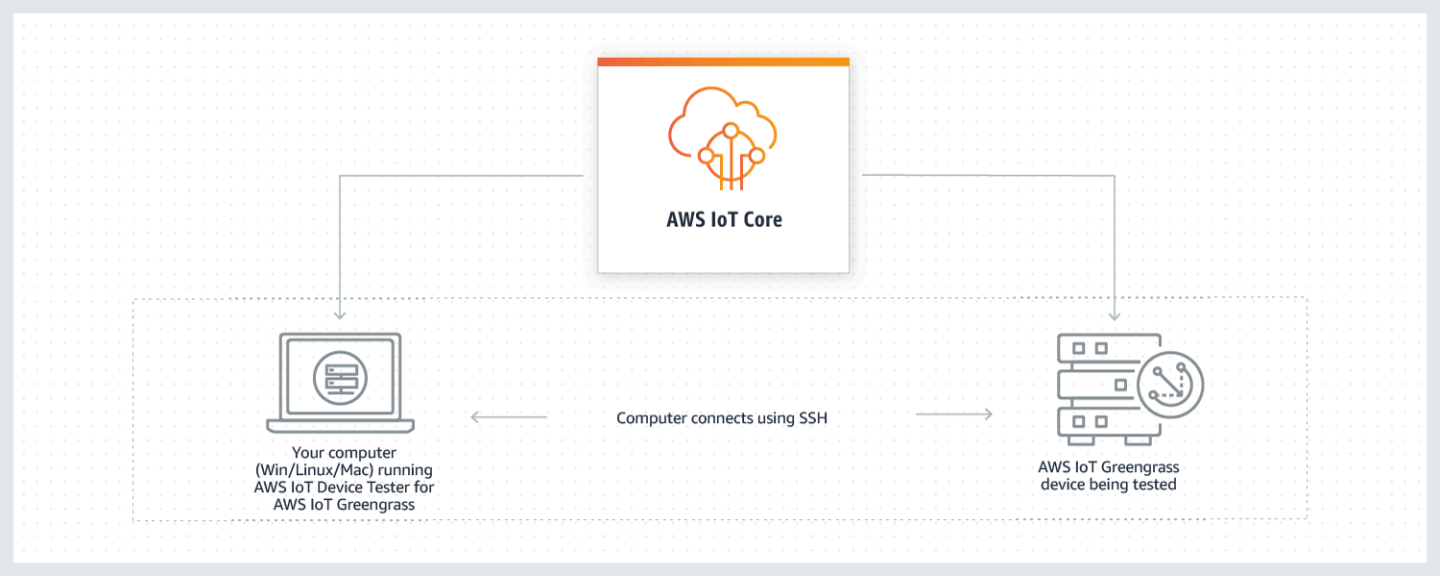
AWS IoT Device Tester (IDT) adalah kerangka pengujian yang dapat diunduh yang memungkinkan Anda memvalidasi perangkat IoT. Anda dapat menggunakan IDT for AWS IoT Greengrass untuk menjalankan rangkaian AWS IoT Greengrass kualifikasi, dan membuat serta menjalankan rangkaian pengujian khusus untuk perangkat Anda.

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 AWS IoT Device Tester untuk AWS IoT Greengrass V2 untuk memverifikasi bahwa perangkat lunak AWS IoT Greengrass Core berjalan pada perangkat keras Anda dan dapat berkomunikasi dengan perangkat keras AWS Cloud. Ini juga melakukan end-to-end tes dengan AWS IoT Core. Misalnya, aplikasi ini memverifikasi bahwa perangkat Anda dapat men-deploy komponen dan memperbaruinya.

Jika Anda ingin menambahkan perangkat keras ke Katalog AWS Partner Perangkat, jalankan rangkaian AWS IoT Greengrass kualifikasi untuk menghasilkan laporan pengujian yang dapat Anda kirimkan AWS IoT. Untuk informasi selengkapnya, lihat Program Kualifikasi Perangkat AWS.



IDT untuk AWS IoT Greengrass V2 mengatur pengujian menggunakan konsep rangkaian pengujian dan kelompok uji.

- Rangkaian uji adalah rangkaian grup uji yang digunakan untuk memverifikasi bahwa perangkat bekerja dengan versi AWS IoT Greengrass tertentu.
- Grup uji adalah serangkaian pengujian individu yang terkait dengan fitur tertentu, seperti deployment komponen.

Untuk informasi selengkapnya, lihat Gunakan IDT untuk menjalankan suite AWS IoT Greengrass kualifikasi.

Rangkaian pengujian khusus

Mulai IDT v4.0.1, IDT untuk AWS IoT Greengrass V2 menggabungkan pengaturan konfigurasi standar dan format hasil dengan lingkungan rangkaian pengujian yang memungkinkan Anda mengembangkan rangkaian pengujian khusus untuk perangkat dan perangkat lunak perangkat Anda. Anda dapat menambahkan tes 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 rangkain tes Anda sendiri.

Versi yang didukung AWS IoT Device Tester untuk AWS IoT Greengrass V2

Topik ini mencantumkan versi IDT yang didukung untuk AWS IoT Greengrass V2. Sebagai praktik terbaik, kami menyarankan Anda menggunakan IDT versi terbaru untuk AWS IoT Greengrass V2 yang mendukung versi target AWS IoT Greengrass V2 Anda. Rilis baru AWS IoT Greengrass mungkin mengharuskan Anda mengunduh versi baru IDT untuk AWS IoT Greengrass V2. Anda menerima pemberitahuan saat memulai uji coba jika IDT untuk AWS IoT Greengrass V2 tidak kompatibel dengan versi yang AWS IoT Greengrass Anda gunakan.

Dengan mengunduh perangkat lunak tersebut, Anda menyetujui Perjanjian Lisensi AWS IoT Device Tester.

> **Note**
>
> IDT tidak mendukung untuk 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.

Versi IDT terbaru untuk V2 AWS IoT Greengrass

Anda dapat menggunakan versi IDT ini untuk AWS IoT Greengrass V2 dengan AWS IoT Greengrass versi yang tercantum di sini.

IDT v4.9.4 untuk AWS IoT Greengrass

AWS IoT Greengrass Versi yang didukung:

- Inti Greengrass v2.12.0, v2.11.0, v2.10.0, dan v2.9.5

Unduhan perangkat lunak IDT:

- IDT v4.9.4 dengan test suite GGV2Q_2.5.4 untuk Linux
- IDT v4.9.4 dengan test suite GGV2Q_2.5.4 untuk macOS
- IDT v4.9.4 dengan test suite GGV2Q_2.5.4 untuk Windows

Catatan rilis:

- Mengaktifkan validasi dan kualifikasi perangkat untuk perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core versi 2.12.0, 2.11.0, 2.10.0, dan 2.9.5.
- Menghapus pengelola aliran dan grup uji pembelajaran mesin.

Catatan tambahan:

- Jika perangkat Anda menggunakan HSM dan Anda menggunakan nucleus 2.10.x, bermigrasi ke Greengrass nucleus versi 2.11.0 atau yang lebih baru.

Versi rangkaian tes:

`GGV2Q_2.5.4`

- Dirilis 2024.05.03

Versi IDT sebelumnya untuk AWS IoT Greengrass

Versi IDT sebelumnya untuk AWS IoT Greengrass V2 berikut juga didukung.

IDT v4.9.3 untuk AWS IoT Greengrass

AWS IoT Greengrass Versi yang didukung:

- Inti Greengrass v2.12.0, v2.11.0, v2.10.0, dan v2.9.5

Unduhan perangkat lunak IDT:

- IDT v4.9.3 dengan test suite GGV2Q_2.5.3 untuk Linux
- IDT v4.9.3 dengan test suite GGV2Q_2.5.3 untuk macOS
- IDT v4.9.3 dengan test suite GGV2Q_2.5.3 untuk Windows

Catatan rilis:

- Memperbaiki masalah dalam pengujian komponen saat menguji perangkat Linux dari host Windows atau sebaliknya.
- Menghapus kasus `localcomponent` uji dari kelompok `component` uji. Kasus uji ini tidak lagi diperlukan untuk kualifikasi.

Catatan tambahan:

- Jika perangkat Anda menggunakan HSM dan Anda menggunakan nucleus 2.10.x, bermigrasi ke Greengrass nucleus versi 2.11.0 atau yang lebih baru.

Versi rangkaian tes:

`GGV2Q_2.5.3`

- Dirilis 2024.04.05

Versi untuk V2 yang tidak AWS IoT Device Tester didukung AWS IoT Greengrass

Topik ini mencantumkan versi IDT yang tidak didukung untuk AWS IoT Greengrass V2. Versi yang tidak didukung tidak menerima perbaikan bug atau pembaruan. Untuk informasi selengkapnya, lihat the section called “Kebijakan Support AWS IoT Device Tester untuk AWS IoT Greengrass”.

IDT v4.9.2 untuk AWS IoT Greengrass

Catatan rilis:

- Memperbaiki masalah di mana rangkaian pengujian Lambda gagal karena Java 8 tidak digunakan lagi.

Versi rangkaian tes:

`GGV2Q_2.5.2`

- Dirilis 2024.03.18

IDT v4.9.1 untuk AWS IoT Greengrass

Catatan rilis:

- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core versi 2.12.0, 2.11.0, 2.10.0, dan 2.9.5.
- Perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_2.5.1`

- Dirilis 2023.10.05

IDT v4.7.0 untuk AWS IoT Greengrass

AWS IoT Greengrass Versi yang didukung:

- Inti Greengrass v2.11.0, v2.10.0, dan v2.9.5

Catatan rilis:

- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core versi 2.11.0, 2.10.0, dan 2.9.5.
- Menambahkan dukungan untuk menyimpan nilai userdata IDT di AWS Systems Manager Parameter Store dan mengambilnya ke dalam konfigurasi menggunakan sintaks placeholder.
- Perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_2.5.0`

- Dirilis 2022.12.13

IDT v4.5.11 untuk AWS IoT Greengrass

Catatan rilis:

- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core versi 2.9.1, 2.9.0, 2.8.1, 2.8.0, 2.7.0, dan 2.6.0.
- Menambahkan dukungan untuk menguji PreInstalled Greengrass pada perangkat inti.
- Perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_2.4.1`

- Dirilis 2022.10.13

IDT v4.5.8 untuk AWS IoT Greengrass

Catatan rilis:

- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core versi 2.7.0, 2.6.0, dan 2.5.6.
- Memungkinkan Anda menguji dengan PreInstalled Greengrass pada perangkat inti.
- Perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_2.4.0`

- Dirilis 2022.08.12

IDT v4.5.3 untuk AWS IoT Greengrass

Catatan rilis:

- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core versi 2.7.0, 2.6.0, 2.5.6, 2.5.5, 2.5.4, dan 2.5.3.
- Update DockerApplicationManager tes untuk menggunakan image docker berbasis ECR-based.
- Perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_2.3.1`

- Dirilis 2022.04.15

IDT v4.5.1 untuk AWS IoT Greengrass

Catatan rilis:

- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core v2.5.3.
- Menambahkan dukungan untuk memvalidasi dan memenuhi syarat perangkat berbasis Linux yang menggunakan modul keamanan perangkat keras (HSM) untuk menyimpan kunci pribadi dan sertifikat yang digunakan oleh perangkat lunak Core. AWS IoT Greengrass
- Mengimplementasikan orkestrator pengujian IDT baru untuk mengonfigurasi rangkaian pengujian kustom. Untuk informasi selengkapnya, lihat Konfigurasikan orkestrasi uji IDT.

- Tambahan perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_2.3.0`

- Dirilis 2022.01.11

IDT v4.4.1 untuk AWS IoT Greengrass

Catatan rilis:

- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core v2.5.2.
- Menambahkan dukungan untuk menggunakan peran IAM yang ditentukan pengguna sebagai peran pertukaran token yang diasumsikan perangkat yang diuji untuk berinteraksi dengan sumber daya. AWS

  Anda dapat menentukan peran IAM dalam `userdata.json`file. Jika Anda menentukan peran kustom, IDT menggunakan peran tersebut alih-alih membuat peran pertukaran token default selama pengujian dijalankan.
- Tambahan perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_2.2.1`

- Dirilis 2021.12.12

IDT v4.4.0 untuk AWS IoT Greengrass

Catatan rilis:

- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core v2.5.0.
- Menambahkan dukungan untuk memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core di Windows.
- Mendukung penggunaan validasi kunci publik untuk koneksi perangkat shell aman (SSH).
- Meningkatkan kebijakan IAM izin IDT dengan praktik terbaik keamanan.
- Tambahan perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_2.1.0`

- Dirilis 2021.11.19

IDT v4.2.0 untuk AWS IoT Greengrass

Catatan rilis:

- Termasuk dukungan untuk kualifikasi fitur berikut pada perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core v2.2.0 dan versi yang lebih baru:
  - Docker—Memvalidasi bahwa perangkat dapat mengunduh gambar kontainer Docker dari Amazon Elastic Container Registry (Amazon ECR).
  - Pembelajaran mesin—memvalidasi bahwa perangkat dapat melakukan inferensi pembelajaran mesin (ML) menggunakan kerangka kerja Deep Learning Runtime atau Lite ML. TensorFlow
  - Stream Manager—memvalidasi bahwa perangkat dapat mengunduh, menginstal, dan menjalankan pengelola aliran. AWS IoT Greengrass
- Memungkinkan Anda memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core v2.4.0, v2.3.0, v2.2.0, dan v2.1.0.
- Kelompokkan log pengujian untuk setiap kasus uji dalam folder *< test-case-id >* terpisah di dalam *<device-tester-extract-location>*`/results/`*<execution-id>*`/logs/`*<test-group-id>* direktori.
- Tambahan perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_2.0.1`

- Dirilis 2021.08.31

IDT v4.1.0 untuk AWS IoT Greengrass

Catatan rilis:

- Memungkinkan Anda untuk memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak inti AWS IoT Greengrass v2.3.0, v2.2.0, v2.1.0, dan v2.0.5.
- Meningkatkan konfigurasi `userdata.json` dengan menghapus persyaratan untuk menentukan properti `GreengrassNucleusVersion` dan `GreengrassCLIVersion`.
- Termasuk dukungan untuk kualifikasi fitur Lambda dan MQTT untuk perangkat lunak AWS IoT Greengrass Core v2.1.0 dan versi yang lebih baru. Anda sekarang dapat menggunakan IDT untuk AWS IoT Greengrass V2 untuk memvalidasi bahwa perangkat inti Anda dapat menjalankan fungsi Lambda dan perangkat dapat mempublikasikan dan berlangganan topik MQTT. AWS IoT Core
- Meningkatkan kemampuan pencatatan.

- Tambahan perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_1.1.1`

- Dirilis 2021.06.18

IDT v4.0.2 untuk AWS IoT Greengrass

Catatan rilis:

- Memungkinkan Anda untuk memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak inti AWS IoT Greengrass v2.1.0.
- Menambahkan dukungan untuk kualifikasi fitur Lambda dan MQTT untuk perangkat lunak AWS IoT Greengrass Core v2.1.0 dan versi yang lebih baru. Anda sekarang dapat menggunakan IDT untuk AWS IoT Greengrass V2 untuk memvalidasi bahwa perangkat inti Anda dapat menjalankan fungsi Lambda dan perangkat dapat mempublikasikan dan berlangganan topik MQTT. AWS IoT Core
- Meningkatkan kemampuan pencatatan.
- Tambahan perbaikan bug minor.

Versi rangkaian tes:

`GGV2Q_1.1.1`

- Dirilis 2021.05.05

IDT v4.0.1 untuk AWS IoT Greengrass

Catatan rilis:

- Memungkinkan Anda untuk memvalidasi dan memenuhi syarat perangkat yang menjalankan perangkat lunak inti AWS IoT Greengrass Versi 2.
- Memungkinkan Anda mengembangkan dan menjalankan suite pengujian kustom Anda menggunakan AWS IoT Device Tester for AWS IoT Greengrass. Untuk informasi selengkapnya, lihat Gunakan IDT untuk mengembangkan dan menjalankan rangkain tes Anda sendiri.
- Menyediakan aplikasi IDT bertanda tangan kode untuk macOS dan Windows. Di macOS, Anda mungkin harus memberikan pengecualian keamanan untuk IDT. Untuk informasi selengkapnya, lihat Pengecualian keamanan di macOS.

Versi rangkaian tes:

`GGV2Q_1.0.0`

- Dirilis 2020.12.22

- Rangkaian tes ini hanya menjalankan tes yang diperlukan untuk kualifikasi, kecuali Anda mengatur `value` di rangkaian `features` ke `yes`.

Unduh IDT untuk V2 AWS IoT Greengrass

Topik ini menjelaskan opsi untuk mengunduh AWS IoT Device Tester untuk AWS IoT Greengrass V2. Anda dapat menggunakan salah satu tautan unduhan perangkat lunak berikut atau Anda dapat mengikuti petunjuk untuk mengunduh IDT secara terprogram.

Topik

Dengan mengunduh perangkat lunak tersebut, Anda menyetujui Perjanjian Lisensi AWS IoT Device Tester.

> **Note**
>
> IDT tidak mendukung untuk 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.

Unduh IDT secara manual

Topik ini mencantumkan versi IDT yang didukung untuk AWS IoT Greengrass V2. Sebagai praktik terbaik, kami menyarankan Anda menggunakan IDT versi terbaru untuk AWS IoT Greengrass V2 yang mendukung versi target AWS IoT Greengrass V2 Anda. Rilis baru AWS IoT Greengrass mungkin mengharuskan Anda mengunduh versi baru IDT untuk AWS IoT Greengrass V2. Anda menerima pemberitahuan saat memulai uji coba jika IDT untuk AWS IoT Greengrass V2 tidak kompatibel dengan versi yang AWS IoT Greengrass Anda gunakan.

IDT v4.9.4 untuk AWS IoT Greengrass

AWS IoT Greengrass Versi yang didukung:

- Inti Greengrass v2.12.0, v2.11.0, v2.10.0, dan v2.9.5

Unduhan perangkat lunak IDT:

- IDT v4.9.4 dengan test suite GGV2Q_2.5.4 untuk Linux
- IDT v4.9.4 dengan test suite GGV2Q_2.5.4 untuk macOS
- IDT v4.9.4 dengan test suite GGV2Q_2.5.4 untuk Windows

Catatan rilis:

- Mengaktifkan validasi dan kualifikasi perangkat untuk perangkat yang menjalankan perangkat lunak AWS IoT Greengrass Core versi 2.12.0, 2.11.0, 2.10.0, dan 2.9.5.
- Menghapus pengelola aliran dan grup uji pembelajaran mesin.

Catatan tambahan:

- Jika perangkat Anda menggunakan HSM dan Anda menggunakan nucleus 2.10.x, bermigrasi ke Greengrass nucleus versi 2.11.0 atau yang lebih baru.

Versi rangkaian tes:

`GGV2Q_2.5.4`

- Dirilis 2024.05.03

Unduh IDT secara terprogram

IDT menyediakan operasi API yang dapat Anda gunakan untuk mengambil URL tempat Anda dapat mengunduh IDT secara terprogram. Anda juga dapat menggunakan operasi API ini untuk memeriksa apakah Anda memiliki IDT versi terbaru. Operasi API ini memiliki titik akhir berikut.

```
https://download.devicetester.iotdevicesecosystem.amazonaws.com/latestidt
```

Untuk memanggil operasi API ini, Anda harus memiliki izin untuk melakukan **`iot-device-tester:LatestIdt`** tindakan. Sertakan AWS tanda tangan Anda dan gunakan `iot-device-tester` sebagai nama layanan.

Permintaan API

HostOs — Sistem operasi mesin host. Pilih dari salah satu pilihan berikut:

- `mac`
- `linux`
- `windows`

TestSuiteType — Jenis test suite. Pilih opsi berikut:

`GGV2`- IDT untuk V2 AWS IoT Greengrass

ProductVersion

(Opsional) Versi inti Greengrass. Layanan mengembalikan versi IDT terbaru yang kompatibel untuk versi inti Greengrass tersebut. Jika Anda tidak menentukan opsi ini, layanan akan mengembalikan IDT versi terbaru.

Respon API

Respons API memiliki format berikut. `DownloadURL`Termasuk file zip.

```
{
    "Success": True or False,
    "Message": Message,
    "LatestBk": {
        "Version": The version of the IDT binary,
        "TestSuiteVersion": The version of the test suite,
        "DownloadURL": The URL to download the IDT Bundle, valid for one hour
    }
 }
```

Contoh

Anda dapat mereferensikan contoh-contoh berikut untuk mengunduh IDT secara terprogram. Contoh-contoh ini menggunakan kredensil yang Anda simpan dalam variabel `AWS_ACCESS_KEY_ID` dan `AWS_SECRET_ACCESS_KEY` lingkungan. Untuk mengikuti praktik keamanan terbaik, jangan simpan kredensil Anda dalam kode Anda.

Example Contoh: Unduh menggunakan cURL versi 7.75.0 atau yang lebih baru (Mac dan Linux)

Jika Anda memiliki cURL versi 7.75.0 atau yang lebih baru, Anda dapat menggunakan `aws-sigv4` flag untuk menandatangani permintaan API. Contoh ini menggunakan jq untuk mengurai URL unduhan dari respons.

> **Warning**
>
> `aws-sigv4`Bendera mensyaratkan parameter kueri dari permintaan GET curl berada dalam urutan HostOs/ProductVersion/TestSuiteType atauHostOs/TestSuiteType. Pesanan yang

tidak sesuai, akan mengakibatkan kesalahan mendapatkan tanda tangan yang tidak cocok untuk String Canonical dari API Gateway.
Jika parameter opsional ProductVersion disertakan, Anda harus menggunakan versi produk yang didukung seperti yang didokumentasikan dalam versi yang didukung AWS IoT Device Tester untuk AWS IoT Greengrass V2.

- Ganti *us-west-2* dengan Anda. Wilayah AWS Untuk daftar kode Region, lihat Titik akhir Regional.
- Ganti *linux* dengan sistem operasi mesin host Anda.
- Ganti *2.5.3* dengan versi nukleus Anda. AWS IoT Greengrass

```
url=$(curl --request GET "https://
download.devicetester.iotdevicesecosystem.amazonaws.com/latestidt?
HostOs=linux&ProductVersion=2.5.3&TestSuiteType=GGV2" \
--user $AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY \
--aws-sigv4 "aws:amz:us-west-2:iot-device-tester" \
| jq -r '.LatestBk["DownloadURL"]')

curl $url --output devicetester.zip
```

Example Contoh: Unduh menggunakan cURL versi sebelumnya (Mac dan Linux)

Anda dapat menggunakan perintah cURL berikut dengan AWS tanda tangan yang Anda tanda tangani dan hitung. Untuk informasi selengkapnya tentang cara menandatangani dan menghitung AWS tanda tangan, lihat Menandatangani permintaan AWS API.

- Ganti *linux* dengan sistem operasi mesin host Anda.
- Ganti *Timestamp* dengan tanggal dan waktu, seperti. **20220210T004606Z**
- Ganti *Tanggal* dengan tanggal, seperti**20220210**.
- Ganti *AWSRegion*dengan Anda Wilayah AWS. Untuk daftar kode Region, lihat Titik akhir Regional.
- Ganti *AWSSignature*dengan AWS tanda tangan yang Anda hasilkan.

```
curl --location --request GET 'https://
download.devicetester.iotdevicesecosystem.amazonaws.com/latestidt?
HostOs=linux&TestSuiteType=GGV2' \
--header 'X-Amz-Date: Timestamp \
```

```
--header 'Authorization: AWS4-HMAC-SHA256 Credential=$AWS_ACCESS_KEY_ID/Date/AWSRegion/
iot-device-tester/aws4_request, SignedHeaders=host;x-amz-date, Signature=AWSSignature'
```

Example Contoh: Unduh menggunakan skrip Python

Contoh ini menggunakan pustaka permintaan Python. Contoh ini diadaptasi dari contoh Python untuk Menandatangani permintaan AWS API di Referensi AWS Umum.

- Ganti *us-west-2* dengan Wilayah Anda. Untuk daftar kode Region, lihat Titik akhir Regional.
- Ganti *linux* dengan sistem operasi mesin host Anda.

```
# Copyright 2010-2022 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
#License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# See: http://docs.aws.amazon.com/general/latest/gr/sigv4_signing.html
# This version makes a GET request and passes the signature
# in the Authorization header.
import sys, os, base64, datetime, hashlib, hmac
import requests # pip install requests
# ************* REQUEST VALUES *************
method = 'GET'
service = 'iot-device-tester'
host = 'download.devicetester.iotdevicesecosystem.amazonaws.com'
region = 'us-west-2'
endpoint = 'https://download.devicetester.iotdevicesecosystem.amazonaws.com/latestidt'
request_parameters = 'HostOs=linux&TestSuiteType=GGV2'

# Key derivation functions. See:
# http://docs.aws.amazon.com/general/latest/gr/signature-v4-examples.html#signature-v4-
examples-python
def sign(key, msg):
    return hmac.new(key, msg.encode('utf-8'), hashlib.sha256).digest()
```

```
def getSignatureKey(key, dateStamp, regionName, serviceName):
    kDate = sign(('AWS4' + key).encode('utf-8'), dateStamp)
    kRegion = sign(kDate, regionName)
    kService = sign(kRegion, serviceName)
    kSigning = sign(kService, 'aws4_request')
    return kSigning

# Read AWS access key from env. variables or configuration file. Best practice is NOT
# to embed credentials in code.
access_key = os.environ.get('AWS_ACCESS_KEY_ID')
secret_key = os.environ.get('AWS_SECRET_ACCESS_KEY')
if access_key is None or secret_key is None:
    print('No access key is available.')
    sys.exit()

# Create a date for headers and the credential string
t = datetime.datetime.utcnow()
amzdate = t.strftime('%Y%m%dT%H%M%SZ')
datestamp = t.strftime('%Y%m%d') # Date w/o time, used in credential scope

# ************* TASK 1: CREATE A CANONICAL REQUEST *************
# http://docs.aws.amazon.com/general/latest/gr/sigv4-create-canonical-request.html
# Step 1 is to define the verb (GET, POST, etc.)--already done.
# Step 2: Create canonical URI--the part of the URI from domain to query
# string (use '/' if no path)
canonical_uri = '/latestidt'
# Step 3: Create the canonical query string. In this example (a GET request),
# request parameters are in the query string. Query string values must
# be URL-encoded (space=%20). The parameters must be sorted by name.
# For this example, the query string is pre-formatted in the request_parameters
 variable.
canonical_querystring = request_parameters
# Step 4: Create the canonical headers and signed headers. Header names
# must be trimmed and lowercase, and sorted in code point order from
# low to high. Note that there is a trailing \n.
canonical_headers = 'host:' + host + '\n' + 'x-amz-date:' + amzdate + '\n'
# Step 5: Create the list of signed headers. This lists the headers
# in the canonical_headers list, delimited with ";" and in alpha order.
# Note: The request can include any headers; canonical_headers and
# signed_headers lists those that you want to be included in the
# hash of the request. "Host" and "x-amz-date" are always required.
signed_headers = 'host;x-amz-date'
# Step 6: Create payload hash (hash of the request body content). For GET
# requests, the payload is an empty string ("").
```

```
payload_hash = hashlib.sha256(('').encode('utf-8')).hexdigest()
# Step 7: Combine elements to create canonical request
canonical_request = method + '\n' + canonical_uri + '\n' + canonical_querystring + '\n'
 + canonical_headers + '\n' + signed_headers + '\n' + payload_hash

# ************* TASK 2: CREATE THE STRING TO SIGN*************
# Match the algorithm to the hashing algorithm you use, either SHA-1 or
# SHA-256 (recommended)
algorithm = 'AWS4-HMAC-SHA256'
credential_scope = datestamp + '/' + region + '/' + service + '/' + 'aws4_request'
string_to_sign = algorithm + '\n' +  amzdate + '\n' +  credential_scope + '\n' +
 hashlib.sha256(canonical_request.encode('utf-8')).hexdigest()
# ************* TASK 3: CALCULATE THE SIGNATURE *************
# Create the signing key using the function defined above.
signing_key = getSignatureKey(secret_key, datestamp, region, service)
# Sign the string_to_sign using the signing_key
signature = hmac.new(signing_key, (string_to_sign).encode('utf-8'),
 hashlib.sha256).hexdigest()

# ************* TASK 4: ADD SIGNING INFORMATION TO THE REQUEST *************
# The signing information can be either in a query string value or in
# a header named Authorization. This code shows how to use a header.
# Create authorization header and add to request headers
authorization_header = algorithm + ' ' + 'Credential=' + access_key + '/' +
 credential_scope + ', ' +  'SignedHeaders=' + signed_headers + ', ' + 'Signature=' +
 signature
# The request can include any headers, but MUST include "host", "x-amz-date",
# and (for this scenario) "Authorization". "host" and "x-amz-date" must
# be included in the canonical_headers and signed_headers, as noted
# earlier. Order here is not significant.
# Python note: The 'host' header is added automatically by the Python 'requests'
 library.
headers = {'x-amz-date':amzdate, 'Authorization':authorization_header}

# ************* SEND THE REQUEST *************
request_url = endpoint + '?' + canonical_querystring
print('\nBEGIN REQUEST++++++++++++++++++++++++++++++++++++')
print('Request URL = ' + request_url)
response = requests.get(request_url, headers=headers)
print('\nRESPONSE++++++++++++++++++++++++++++++++++++')
print('Response code: %d\n' % response.status_code)
print(response.text)

download_url = response.json()["LatestBk"]["DownloadURL"]
```

```
r = requests.get(download_url)
open('devicetester.zip', 'wb').write(r.content)
```

Gunakan IDT untuk menjalankan suite AWS IoT Greengrass kualifikasi

Anda dapat menggunakan AWS IoT Greengrass V2 AWS IoT Device Tester untuk memverifikasi bahwa perangkat lunak AWS IoT Greengrass Core berjalan pada perangkat keras Anda dan dapat berkomunikasi dengan perangkat keras AWS Cloud. Ini juga melakukan end-to-end tes dengan AWS IoT Core. Misalnya, aplikasi ini memverifikasi bahwa perangkat Anda dapat men-deploy komponen dan memperbaruinya.

Selain perangkat pengujian, IDT untuk AWS IoT Greengrass V2 menciptakan sumber daya (misalnya, AWS IoT hal-hal, grup, dan sebagainya) di Anda Akun AWS untuk memfasilitasi proses kualifikasi.

Untuk membuat sumber daya ini, IDT untuk AWS IoT Greengrass V2 menggunakan AWS kredenal yang dikonfigurasi dalam `config.json` file untuk melakukan panggilan API atas nama Anda. Sumber daya ini disediakan pada berbagai waktu selama tes.

Ketika Anda menggunakan IDT untuk AWS IoT Greengrass V2 untuk menjalankan suite AWS IoT Greengrass kualifikasi, ia melakukan langkah-langkah berikut:

1. Memuat dan memvalidasi konfigurasi perangkat dan 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 forum Anda lulus tes yang diperlukan untuk kualifikasi.

Versi rangkaian tes

IDT untuk AWS IoT Greengrass V2 mengatur pengujian ke dalam rangkaian pengujian dan grup pengujian.

- Rangkaian uji adalah rangkaian grup uji yang digunakan untuk memverifikasi bahwa perangkat bekerja dengan versi AWS IoT Greengrass tertentu.

- Grup uji adalah serangkaian pengujian individu yang terkait dengan fitur tertentu, seperti deployment komponen.

Rangkaian uji dibuat dalam versi baru dengan menggunakan format *major*.*minor*.*patch*, misalnya `GGV2Q_1.0.0`. Ketika Anda men-download IDT, paketnya mencakup versi terbaru rangkaian Greengrass kualifikasi.

> **Important**
>
> Pengujian dari versi rangkaian uji yang tidak didukung tidak valid untuk kualifikasi perangkat. IDT tidak mencetak laporan kualifikasi untuk versi yang tidak didukung. Untuk informasi selengkapnya, lihat the section called “Kebijakan Support AWS IoT Device Tester untuk AWS IoT Greengrass”.
>
> Anda dapat `list-supported-products` menjalankan daftar versi AWS IoT Greengrass dan test suite yang didukung oleh versi IDT Anda saat ini.

Deskripsi grup uji

Kelompok Uji yang Diperlukan untuk Kualifikasi Inti

Grup pengujian ini diperlukan untuk memenuhi syarat perangkat AWS IoT Greengrass V2 Anda untuk Katalog AWS Partner Perangkat.

Dependensi Inti

Memvalidasi bahwa perangkat tersebut memenuhi semua persyaratan perangkat lunak dan perangkat keras untuk perangkat lunak inti AWS IoT Greengrass . Grup uji ini mencakup uji kasus berikut:

Versi Java

Memeriksa apakah versi Java yang diperlukan diinstal pada perangkat yang sedang diuji. AWS IoT Greengrass membutuhkan Java 8 atau yang lebih baru.

PreTest Validasi

Memeriksa apakah perangkat memenuhi persyaratan perangkat lunak untuk menjalankan pengujian.

- Untuk perangkat berbasis Linux, pengujian ini memeriksa apakah perangkat dapat menjalankan perintah Linux berikut:

`chmod`, `cp`, `echo`, `grep`, `kill`, `ln`, `mkinfo`, `ps`, `rm`, `sh`, `uname`

- Untuk perangkat berbasis Windows, pengujian ini memeriksa apakah perangkat telah menginstal perangkat lunak Microsoft berikut:

  Powershell v5.1 atau yang lebih baru, .NET v4.6.1 atau yang lebih baru, Visual C++ 2017 atau yang lebih baru, utilitas PsExec

Pemeriksa Versi

Memeriksa apakah versi yang AWS IoT Greengrass disediakan kompatibel dengan versi AWS IoT Device Tester yang Anda gunakan.

Komponen

Memvalidasi bahwa perangkat dapat men-deploy komponen dan meningkatkannya. Grup uji ini mencakup pengujian berikut:

Komponen Cloud

Memvalidasi kemampuan perangkat untuk komponen cloud.

Komponen Lokal

Memvalidasi kemampuan perangkat untuk komponen cloud.

Lambda

Tes ini tidak berlaku untuk perangkat berbasis Windows.

Memvalidasi bahwa perangkat dapat menerapkan komponen fungsi Lambda yang menggunakan runtime Java, dan bahwa fungsi Lambda dapat menggunakan topik AWS IoT Core MQTT sebagai sumber peristiwa untuk pesan kerja.

MQTT

Memvalidasi bahwa perangkat dapat berlangganan dan mempublikasikan ke topik AWS IoT Core MQTT.

Grup Uji Opsional

> **Note**
>
> Grup pengujian ini bersifat opsional, dan hanya digunakan untuk perangkat inti Greengrass berbasis Linux yang memenuhi syarat. Jika Anda memilih untuk memenuhi

syarat untuk pengujian opsional, perangkat Anda terdaftar dengan kemampuan tambahan di Katalog AWS Partner Perangkat.

Dependensi Docker

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk menggunakan komponen Docker application manager () AWS-provided. `aws.greengrass.DockerApplicationManager`

Kualifikasi Manajer Aplikasi Docker

Memvalidasi bahwa perangkat dapat mengunduh gambar kontainer Docker dari Amazon ECR.

Dependensi Machine Learning

> **Note**
>
> Kelompok uji opsional pembelajaran mesin hanya didukung di IDT v4.9.3.

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk menggunakan komponen pembelajaran mesin (AWS ML) yang disediakan.

Uji Inferensi Machine Learning

> **Note**
>
> Kelompok uji opsional pembelajaran mesin hanya didukung di IDT v4.9.3.

Memvalidasi bahwa perangkat dapat melakukan inferensi ML menggunakan kerangka kerja Deep Learning Runtime dan TensorFlow Lite ML.

Dependensi Stream Manager

> **Note**
>
> Grup pengujian opsional manajer aliran hanya didukung di IDT v4.9.3.

Memvalidasi bahwa perangkat tersebut dapat mengunduh, menginstal, dan menjalankan perintah manajer pengaliran AWS IoT Greengrass.

Integrasi Keamanan Perangkat Keras (HSI)

> **Note**
>
> Tes ini tersedia di IDT v4.9.3 dan yang lebih baru hanya untuk perangkat berbasis Linux. AWS IoT Greengrass saat ini tidak mendukung integrasi keamanan perangkat keras untuk perangkat Windows.

Memvalidasi bahwa perangkat dapat mengautentikasi koneksi ke AWS IoT dan AWS IoT Greengrass layanan menggunakan kunci pribadi dan sertifikat yang disimpan dalam modul keamanan perangkat keras (HSM). Pengujian ini juga memverifikasi bahwa komponen penyedia PKCS #11 AWS yang disediakan dapat berinteraksi dengan HSM menggunakan pustaka PKCS #11 yang disediakan vendor. Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras.

Prasyarat untuk menjalankan suite kualifikasi AWS IoT Greengrass

Bagian ini menjelaskan prasyarat untuk menggunakan AWS IoT Device Tester (IDT) untuk. AWS IoT Greengrass

Unduh versi terbaru AWS IoT Device Tester untuk AWS IoT Greengrass

Unduh versi terbaru IDT dan ekstrak perangkat lunak ke lokasi (*< device-tester-extract-location >*) pada sistem file Anda di mana Anda memiliki izin baca/tulis.

> **Note**
>
> IDT tidak mendukung untuk 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.

Unduh perangkat AWS IoT Greengrass lunak

IDT untuk AWS IoT Greengrass V2 menguji perangkat Anda untuk kompatibilitas dengan versi tertentu. AWS IoT Greengrass Jalankan perintah berikut untuk mengunduh perangkat lunak AWS IoT Greengrass Core ke file bernamaaws.greengrass.nucleus.zip. Ganti *versi* dengan versi komponen nukleus yang didukung untuk versi IDT Anda.

Linux or Unix

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

Windows Command Prompt (CMD)

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

PowerShell

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

Tempatkan file aws.greengrass.nucleus.zip yang sudah diunduh di folder *<device-tester-extract-location>*/products/.

> **Note**
>
> Jangan menempatkan beberapa file dalam direktori ini untuk sistem operasi dan arsitektur yang sama.

Buat dan konfigurasikan Akun AWS

Sebelum Anda dapat menggunakan AWS IoT Device Tester untuk AWS IoT Greengrass V2, Anda harus melakukan langkah-langkah berikut:

1. Mengatur sebuah Akun 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-hal dan AWS IoT Greengrass komponen, atas nama Anda.

Untuk membuat sumber daya ini, IDT untuk AWS IoT Greengrass V2 menggunakan AWS kredensil yang dikonfigurasi dalam `config.json` file untuk melakukan panggilan API atas nama Anda. Sumber daya ini disediakan pada berbagai waktu selama tes.

> **Note**
>
> Meskipun sebagian besar pengujian memenuhi syarat untuk Tingkat Gratis AWS, Anda harus menyediakan kartu kredit saat mendaftar Akun AWS. Untuk informasi selengkapnya, lihat Mengapa saya memerlukan metode pembayaran jika akun saya dilindungi oleh Tingkat Gratis?.

Langkah 1: Siapkan 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”.

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

Untuk mendaftar untuk Akun AWS

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

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

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

Untuk membuat pengguna administrator, pilih salah satu opsi berikut.

Pilih salah satu cara untuk mengelola administrator Anda	Untuk	Oleh	Anda juga bisa
Di Pusat Identitas IAM (Direkomendasikan)	Gunakan kredensi jangka pendek untuk mengakses. AWS Ini sejalan dengan praktik terbaik keamanan. Untuk informasi tentang praktik terbaik, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.	Mengikuti petunjuk di Memulai di Panduan AWS IAM Identity Center Pengguna.	Konfigurasikan akses terprogram dengan Mengonfigurasi AWS CLI yang akan digunakan AWS IAM Identity Center dalam AWS Command Line Interface Panduan Pengguna.
Di IAM (Tidak direkomendasikan)	Gunakan kredensi jangka panjang untuk mengakses. AWS	Mengikuti petunjuk dalam Membuat pengguna admin IAM pertama Anda dan grup pengguna di Panduan Pengguna IAM.	Konfigurasikan akses terprogram dengan Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Langkah 2: Konfigurasikan izin untuk IDT

Pada langkah ini, konfigurasikan izin yang digunakan IDT untuk AWS IoT Greengrass V2 untuk menjalankan pengujian dan mengumpulkan data penggunaan IDT. Anda dapat menggunakanAWS 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 tes untuk IDT, lewati ke Konfigurasikan perangkat Anda untuk menjalankan tes IDT.

Untuk mengonfigurasi izin untuk IDT (konsol)

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. Jika Anda tidak menggunakan PreInstalled, pada tab JSON, ganti konten placeholder dengan kebijakan berikut. Jika Anda menggunakan PreInstalled, lanjutkan ke langkah berikut.

```
{
    "Version":"2012-10-17",
    "Statement":[
    {
      "Sid":"passRoleForResources",
      "Effect":"Allow",
      "Action":"iam:PassRole",
      "Resource":"arn:aws:iam::*:role/idt-*",
      "Condition":{
        "StringEquals":{
          "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid":"lambdaResources",
      "Effect":"Allow",
      "Action":[
        "lambda:CreateFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource":[
```

```
        "arn:aws:lambda:*:*:function:idt-*"
      ]
    },
    {
      "Sid":"iotResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateThing",
        "iot:DeleteThing",
        "iot:DescribeThing",
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:AddThingToThingGroup",
        "iot:RemoveThingFromThingGroup",
        "iot:AttachThingPrincipal",
        "iot:DetachThingPrincipal",
        "iot:UpdateCertificate",
        "iot:DeleteCertificate",
        "iot:CreatePolicy",
        "iot:AttachPolicy",
        "iot:DetachPolicy",
        "iot:DeletePolicy",
        "iot:GetPolicy",
        "iot:Publish",
        "iot:TagResource",
        "iot:ListThingPrincipals",
        "iot:ListAttachedPolicies",
        "iot:ListTargetsForPolicy",
        "iot:ListThingGroupsForThing",
        "iot:ListThingsInThingGroup",
        "iot:CreateJob",
        "iot:DescribeJob",
        "iot:DescribeJobExecution",
        "iot:CancelJob"
      ],
      "Resource":[
        "arn:aws:iot:*:*:thing/idt-*",
        "arn:aws:iot:*:*:thinggroup/idt-*",
        "arn:aws:iot:*:*:policy/idt-*",
        "arn:aws:iot:*:*:cert/*",
        "arn:aws:iot:*:*:topic/idt-*",
        "arn:aws:iot:*:*:job/*"
      ]
```

```
    },
    {
      "Sid":"s3Resources",
      "Effect":"Allow",
      "Action":[
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:ListBucketVersions",
        "s3:DeleteBucket",
        "s3:PutObjectTagging",
        "s3:PutBucketTagging"
      ],
      "Resource":"arn:aws:s3::*:idt-*"
    },
    {
      "Sid":"roleAliasResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateRoleAlias",
        "iot:DescribeRoleAlias",
        "iot:DeleteRoleAlias",
        "iot:TagResource",
        "iam:GetRole"
      ],
      "Resource":[
        "arn:aws:iot:*:*:rolealias/idt-*",
        "arn:aws:iam::*:role/idt-*"
      ]
    },
    {
      "Sid":"idtExecuteAndCollectMetrics",
      "Effect":"Allow",
      "Action":[
        "iot-device-tester:SendMetrics",
        "iot-device-tester:SupportedVersion",
        "iot-device-tester:LatestIdt",
        "iot-device-tester:CheckVersion",
        "iot-device-tester:DownloadTestSuite"
      ],
      "Resource":"*"
```

```
    },
    {
      "Sid":"genericResources",
      "Effect":"Allow",
      "Action":[
        "greengrass:*",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:ListThings",
        "iot:DescribeEndpoint",
        "iot:CreateKeysAndCertificate"
      ],
      "Resource":"*"
    },
    {
      "Sid":"iamResourcesUpdate",
      "Effect":"Allow",
      "Action":[
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:TagRole",
        "iam:TagPolicy",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy"
      ],
      "Resource":[
        "arn:aws:iam::*:role/idt-*",
        "arn:aws:iam::*:policy/idt-*"
      ]
    }
  ]
}
```

c. Jika Anda menggunakan PreInstalled, pada tab JSON, ganti konten placeholder dengan kebijakan berikut. Pastikan Anda:

- Ganti *ThingName dan* ThingGroup dalam `iotResources` pernyataan dengan nama benda dan grup benda yang dibuat selama instalasi Greengrass di perangkat Anda yang sedang diuji (DUT) untuk menambahkan izin.
- Ganti *PassRole dan RoleAlias dalam roleAliasResources pernyataan dan pernyataan dengan peran* `passRoleForResources` yang dibuat selama instalasi Greengrass di DUT Anda.

```
{
    "Version":"2012-10-17",
    "Statement":[
    {
      "Sid":"passRoleForResources",
      "Effect":"Allow",
      "Action":"iam:PassRole",
      "Resource":"arn:aws:iam::*:role/passRole",
      "Condition":{
        "StringEquals":{
          "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid":"lambdaResources",
      "Effect":"Allow",
      "Action":[
        "lambda:CreateFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource":[
        "arn:aws:lambda:*:*:function:idt-*"
      ]
    },
    {
      "Sid":"iotResources",
```

```
      "Effect":"Allow",
      "Action":[
        "iot:CreateThing",
        "iot:DeleteThing",
        "iot:DescribeThing",
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:AddThingToThingGroup",
        "iot:RemoveThingFromThingGroup",
        "iot:AttachThingPrincipal",
        "iot:DetachThingPrincipal",
        "iot:UpdateCertificate",
        "iot:DeleteCertificate",
        "iot:CreatePolicy",
        "iot:AttachPolicy",
        "iot:DetachPolicy",
        "iot:DeletePolicy",
        "iot:GetPolicy",
        "iot:Publish",
        "iot:TagResource",
        "iot:ListThingPrincipals",
        "iot:ListAttachedPolicies",
        "iot:ListTargetsForPolicy",
        "iot:ListThingGroupsForThing",
        "iot:ListThingsInThingGroup",
        "iot:CreateJob",
        "iot:DescribeJob",
        "iot:DescribeJobExecution",
        "iot:CancelJob"
      ],
      "Resource":[
        "arn:aws:iot:*:*:thing/thingName",
        "arn:aws:iot:*:*:thinggroup/thingGroup",
        "arn:aws:iot:*:*:policy/idt-*",
        "arn:aws:iot:*:*:cert/*",
        "arn:aws:iot:*:*:topic/idt-*",
        "arn:aws:iot:*:*:job/*"
      ]
    },
    {
      "Sid":"s3Resources",
      "Effect":"Allow",
      "Action":[
```

```
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:ListBucketVersions",
        "s3:DeleteBucket",
        "s3:PutObjectTagging",
        "s3:PutBucketTagging"
      ],
      "Resource":"arn:aws:s3::*:idt-*"
    },
    {
      "Sid":"roleAliasResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateRoleAlias",
        "iot:DescribeRoleAlias",
        "iot:DeleteRoleAlias",
        "iot:TagResource",
        "iam:GetRole"
      ],
      "Resource":[
        "arn:aws:iot:*:*:rolealias/roleAlias",
        "arn:aws:iam::*:role/idt-*"
      ]
    },
    {
      "Sid":"idtExecuteAndCollectMetrics",
      "Effect":"Allow",
      "Action":[
        "iot-device-tester:SendMetrics",
        "iot-device-tester:SupportedVersion",
        "iot-device-tester:LatestIdt",
        "iot-device-tester:CheckVersion",
        "iot-device-tester:DownloadTestSuite"
      ],
      "Resource":"*"
    },
    {
      "Sid":"genericResources",
      "Effect":"Allow",
      "Action":[
```

```
        "greengrass:*",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:ListThings",
        "iot:DescribeEndpoint",
        "iot:CreateKeysAndCertificate"
      ],
      "Resource":"*"
    },
    {
      "Sid":"iamResourcesUpdate",
      "Effect":"Allow",
      "Action":[
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:TagRole",
        "iam:TagPolicy",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy"
      ],
      "Resource":[
        "arn:aws:iam::*:role/idt-*",
        "arn:aws:iam::*:policy/idt-*"
      ]
    }
  ]
}
```

> **Note**
>
> Jika Anda ingin menggunakan peran IAM kustom sebagai peran pertukaran token untuk perangkat yang sedang diuji, pastikan Anda memperbarui `roleAliasResources` pernyataan dan `passRoleForResources` pernyataan dalam kebijakan untuk mengizinkan sumber daya peran IAM kustom Anda.

d. Pilih Tinjau kebijakan.

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 Atur izin, pilih Lampirkan kebijakan yang ada ke pengguna secara langsung.

      ii. Cari kebijakan IDTGreengrassIAMPermissions yang Anda buat pada 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 file kredensial AWS .

4. Langkah berikutnya: Konfigurasikan perangkat fisik.

Untuk mengonfigurasi izin untuk IDT (AWS CLI)

1. Di komputer Anda, instal dan konfigurasikan AWS CLI jika belum diinstal. Ikuti langkah-langkah di Menginstal AWS CLI di Panduan Pengguna AWS Command Line Interface .

   > **Note**
   >
   > AWS CLI Ini adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan dari shell baris perintah Anda.

2. Buat kebijakan yang dikelola pelanggan yang memberikan izin untuk mengelola IDT dan peran AWS IoT Greengrass .

a. Jika Anda tidak menggunakan PreInstalled, buka editor teks dan simpan konten kebijakan berikut dalam file JSON. Jika Anda menggunakan PreInstalled, lanjutkan ke langkah berikut.

```
{
    "Version":"2012-10-17",
    "Statement":[
    {
      "Sid":"passRoleForResources",
      "Effect":"Allow",
      "Action":"iam:PassRole",
      "Resource":"arn:aws:iam::*:role/idt-*",
      "Condition":{
        "StringEquals":{
          "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid":"lambdaResources",
      "Effect":"Allow",
      "Action":[
        "lambda:CreateFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource":[
        "arn:aws:lambda:*:*:function:idt-*"
      ]
    },
    {
      "Sid":"iotResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateThing",
        "iot:DeleteThing",
        "iot:DescribeThing",
        "iot:CreateThingGroup",
```

```
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:AddThingToThingGroup",
        "iot:RemoveThingFromThingGroup",
        "iot:AttachThingPrincipal",
        "iot:DetachThingPrincipal",
        "iot:UpdateCertificate",
        "iot:DeleteCertificate",
        "iot:CreatePolicy",
        "iot:AttachPolicy",
        "iot:DetachPolicy",
        "iot:DeletePolicy",
        "iot:GetPolicy",
        "iot:Publish",
        "iot:TagResource",
        "iot:ListThingPrincipals",
        "iot:ListAttachedPolicies",
        "iot:ListTargetsForPolicy",
        "iot:ListThingGroupsForThing",
        "iot:ListThingsInThingGroup",
        "iot:CreateJob",
        "iot:DescribeJob",
        "iot:DescribeJobExecution",
        "iot:CancelJob"
      ],
      "Resource":[
        "arn:aws:iot:*:*:thing/idt-*",
        "arn:aws:iot:*:*:thinggroup/idt-*",
        "arn:aws:iot:*:*:policy/idt-*",
        "arn:aws:iot:*:*:cert/*",
        "arn:aws:iot:*:*:topic/idt-*",
        "arn:aws:iot:*:*:job/*"
      ]
    },
    {
      "Sid":"s3Resources",
      "Effect":"Allow",
      "Action":[
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:CreateBucket",
        "s3:ListBucket",
```

```
        "s3:ListBucketVersions",
        "s3:DeleteBucket",
        "s3:PutObjectTagging",
        "s3:PutBucketTagging"
      ],
      "Resource":"arn:aws:s3::*:idt-*"
    },
    {
      "Sid":"roleAliasResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateRoleAlias",
        "iot:DescribeRoleAlias",
        "iot:DeleteRoleAlias",
        "iot:TagResource",
        "iam:GetRole"
      ],
      "Resource":[
        "arn:aws:iot:*:*:rolealias/idt-*",
        "arn:aws:iam::*:role/idt-*"
      ]
    },
    {
      "Sid":"idtExecuteAndCollectMetrics",
      "Effect":"Allow",
      "Action":[
        "iot-device-tester:SendMetrics",
        "iot-device-tester:SupportedVersion",
        "iot-device-tester:LatestIdt",
        "iot-device-tester:CheckVersion",
        "iot-device-tester:DownloadTestSuite"
      ],
      "Resource":"*"
    },
    {
      "Sid":"genericResources",
      "Effect":"Allow",
      "Action":[
        "greengrass:*",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:ListThings",
        "iot:DescribeEndpoint",
        "iot:CreateKeysAndCertificate"
```

```
      ],
      "Resource":"*"
    },
    {
      "Sid":"iamResourcesUpdate",
      "Effect":"Allow",
      "Action":[
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:TagRole",
        "iam:TagPolicy",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy"
      ],
      "Resource":[
        "arn:aws:iam::*:role/idt-*",
        "arn:aws:iam::*:policy/idt-*"
      ]
    }
  ]
}
```

b. Jika Anda menggunakan PreInstalled, buka editor teks dan simpan konten kebijakan berikut dalam file JSON. Pastikan Anda:

- Ganti *`ThingName dan`* ThingGroup dalam `iotResources` pernyataan yang dibuat selama instalasi Greengrass pada perangkat Anda yang sedang diuji (DUT) untuk menambahkan izin.
- Ganti *`PassRole dan RoleAlias dalam roleAliasResources pernyataan dan pernyataan dengan peran`* `passRoleForResources` yang dibuat selama instalasi Greengrass di DUT Anda.

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

```
      "Sid":"passRoleForResources",
      "Effect":"Allow",
      "Action":"iam:PassRole",
      "Resource":"arn:aws:iam::*:role/passRole",
      "Condition":{
        "StringEquals":{
          "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid":"lambdaResources",
      "Effect":"Allow",
      "Action":[
        "lambda:CreateFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource":[
        "arn:aws:lambda:*:*:function:idt-*"
      ]
    },
    {
      "Sid":"iotResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateThing",
        "iot:DeleteThing",
        "iot:DescribeThing",
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:AddThingToThingGroup",
        "iot:RemoveThingFromThingGroup",
        "iot:AttachThingPrincipal",
        "iot:DetachThingPrincipal",
        "iot:UpdateCertificate",
        "iot:DeleteCertificate",
        "iot:CreatePolicy",
```

```
        "iot:AttachPolicy",
        "iot:DetachPolicy",
        "iot:DeletePolicy",
        "iot:GetPolicy",
        "iot:Publish",
        "iot:TagResource",
        "iot:ListThingPrincipals",
        "iot:ListAttachedPolicies",
        "iot:ListTargetsForPolicy",
        "iot:ListThingGroupsForThing",
        "iot:ListThingsInThingGroup",
        "iot:CreateJob",
        "iot:DescribeJob",
        "iot:DescribeJobExecution",
        "iot:CancelJob"
      ],
      "Resource":[
        "arn:aws:iot:*:*:thing/thingName",
        "arn:aws:iot:*:*:thinggroup/thingGroup",
        "arn:aws:iot:*:*:policy/idt-*",
        "arn:aws:iot:*:*:cert/*",
        "arn:aws:iot:*:*:topic/idt-*",
        "arn:aws:iot:*:*:job/*"
      ]
    },
    {
      "Sid":"s3Resources",
      "Effect":"Allow",
      "Action":[
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:ListBucketVersions",
        "s3:DeleteBucket",
        "s3:PutObjectTagging",
        "s3:PutBucketTagging"
      ],
      "Resource":"arn:aws:s3::*:idt-*"
    },
    {
      "Sid":"roleAliasResources",
```

```
      "Effect":"Allow",
      "Action":[
        "iot:CreateRoleAlias",
        "iot:DescribeRoleAlias",
        "iot:DeleteRoleAlias",
        "iot:TagResource",
        "iam:GetRole"
      ],
      "Resource":[
        "arn:aws:iot:*:*:rolealias/roleAlias",
        "arn:aws:iam::*:role/idt-*"
      ]
    },
    {
      "Sid":"idtExecuteAndCollectMetrics",
      "Effect":"Allow",
      "Action":[
        "iot-device-tester:SendMetrics",
        "iot-device-tester:SupportedVersion",
        "iot-device-tester:LatestIdt",
        "iot-device-tester:CheckVersion",
        "iot-device-tester:DownloadTestSuite"
      ],
      "Resource":"*"
    },
    {
      "Sid":"genericResources",
      "Effect":"Allow",
      "Action":[
        "greengrass:*",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:ListThings",
        "iot:DescribeEndpoint",
        "iot:CreateKeysAndCertificate"
      ],
      "Resource":"*"
    },
    {
      "Sid":"iamResourcesUpdate",
      "Effect":"Allow",
      "Action":[
        "iam:CreateRole",
        "iam:DeleteRole",
```

```
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:TagRole",
        "iam:TagPolicy",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy"
      ],
      "Resource":[
        "arn:aws:iam::*:role/idt-*",
        "arn:aws:iam::*:policy/idt-*"
      ]
    }
  ]
}
```

> **Note**
>
> Jika Anda ingin menggunakan peran IAM kustom sebagai peran pertukaran token untuk perangkat yang sedang diuji, pastikan Anda memperbarui `roleAliasResources` pernyataan dan `passRoleForResources` pernyataan dalam kebijakan untuk mengizinkan sumber daya peran IAM kustom Anda.

c. Jalankan perintah berikut untuk membuat kebijakan terkelola pelanggan bernama `IDTGreengrassIAMPermissions`. Ganti *`policy.json`* dengan path lengkap ke file JSON yang Anda buat di langkah sebelumnya.

```
aws iam create-policy --policy-name IDTGreengrassIAMPermissions --policy-
document file://policy.json
```

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 Anda Akun AWS.

```
aws iam attach-user-policy --user-name IDTGreengrassUser --policy-arn
 arn:aws:iam::<account-id>:policy/IDTGreengrassIAMPermissions
```

4. Buat secret access key untuk pengguna tersebut.

   ```
   aws iam create-access-key --user-name IDTGreengrassUser
   ```

   Simpan output tersebut di lokasi yang aman. Anda menggunakan informasi ini nanti untuk mengonfigurasi file AWS kredensi Anda.

5. Langkah berikutnya: Konfigurasikan perangkat fisik.

AWS IoT Device Tester izin

Kebijakan berikut menjelaskan AWS IoT Device Tester izin.

AWS IoT Device Tester memerlukan izin ini untuk memeriksa versi dan fitur pembaruan otomatis.

- `iot-device-tester:SupportedVersion`

  Memberikan AWS IoT Device Tester izin untuk mengambil daftar produk yang didukung, rangkaian pengujian, dan versi IDT.

- `iot-device-tester:LatestIdt`

  Memberikan AWS IoT Device Tester izin untuk mengambil versi IDT terbaru yang tersedia untuk diunduh.

- `iot-device-tester:CheckVersion`

  Memberikan AWS IoT Device Tester izin untuk memeriksa kompatibilitas versi untuk IDT, suite pengujian, dan produk.

- `iot-device-tester:DownloadTestSuite`

  Memberikan AWS IoT Device Tester izin untuk mengunduh pembaruan suite pengujian.

AWS IoT Device Tester juga menggunakan izin berikut untuk pelaporan metrik opsional:

- `iot-device-tester:SendMetrics`

Memberikan izin AWS untuk mengumpulkan metrik tentang penggunaan AWS IoT Device Tester internal. Jika izin ini dihilangkan, metrik ini tidak akan dikumpulkan.

Konfigurasikan perangkat Anda untuk menjalankan tes IDT

Untuk mengaktifkan IDT untuk menjalankan tes untuk kualifikasi perangkat, Anda harus mengonfigurasi komputer host Anda untuk mengakses perangkat Anda, dan mengonfigurasi izin pengguna pada perangkat Anda.

Instal Java pada komputer host

Dimulai dengan IDT v4.2.0, tes kualifikasi opsional untuk AWS IoT Greengrass memerlukan Java untuk dijalankan.

Anda dapat menggunakan Java versi 8 atau yang lebih besar. Kami menyarankan Anda menggunakan versi dukungan jangka panjang Amazon Corretto atau OpenJDK. Versi 8 atau lebih tinggi diperlukan..

Konfigurasikan komputer host Anda untuk mengakses perangkat yang sedang diuji

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. Sebagai contoh:

   $ ssh-copy-id ***<remote-ssh-user>***@***<remote-device-ip>***

   Di *`remote-ssh-user`*mana nama pengguna yang digunakan untuk masuk ke perangkat Anda yang sedang diuji dan *`remote-device-ip`*merupakan alamat IP perangkat yang diuji untuk menjalankan pengujian. Sebagai contoh:

   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 kredensional pengguna untuk perangkat Windows

Untuk memenuhi syarat perangkat berbasis Windows, Anda harus mengonfigurasi kredensional pengguna di LocalSystem akun pada perangkat yang sedang diuji untuk pengguna berikut:

- Pengguna Greengrass default (). `ggc_user`
- Pengguna yang Anda gunakan untuk terhubung ke perangkat yang sedang diuji. Anda mengkonfigurasi pengguna ini dalam `device.json`file.

Anda harus membuat setiap pengguna di LocalSystem akun pada perangkat yang sedang diuji, dan kemudian menyimpan nama pengguna dan kata sandi untuk pengguna dalam contoh Credential Manager untuk LocalSystem akun tersebut.

Untuk mengkonfigurasi pengguna di perangkat Windows

1. Buka Windows Command Prompt (`cmd.exe`) sebagai administrator.
2. Buat pengguna di LocalSystem akun di perangkat Windows. Jalankan perintah berikut untuk setiap pengguna yang ingin Anda buat. *Untuk pengguna Greengrass default, ganti nama pengguna dengan.* `ggc_user` Ganti *kata sandi* dengan kata sandi yang aman.

   ```
   net user /add user-name password
   ```
3. Unduh dan instal PsExecutilitas dari Microsoft pada perangkat.
4. Gunakan PsExec utilitas untuk menyimpan nama pengguna dan kata sandi untuk pengguna default dalam contoh Credential Manager untuk LocalSystem akun tersebut.

   Jalankan perintah berikut untuk setiap pengguna yang ingin Anda konfigurasi di Credential Manager. *Untuk pengguna Greengrass default, ganti nama pengguna dengan.* `ggc_user` Ganti *kata sandi* dengan kata sandi pengguna yang Anda tetapkan sebelumnya.

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

   Jika PsExec License Agreementterbuka, pilih Acceptuntuk menyetujui lisensi dan jalankan perintah.

> **Note**
>
> Pada perangkat Windows, LocalSystem akun menjalankan inti Greengrass, dan Anda harus menggunakan utilitas untuk menyimpan informasi PsExec pengguna di akun. LocalSystem Menggunakan aplikasi Credential Manager menyimpan informasi ini di akun Windows dari pengguna yang saat ini masuk, bukan LocalSystem akun.

Konfigurasikan izin pengguna di perangkat Anda

IDT melakukan operasi pada berbagai direktori dan file dalam perangkat yang diuji. Beberapa operasi ini memerlukan izin yang ditinggikan (menggunakan sudo). Untuk mengotomatiskan operasi ini, IDT untuk AWS IoT Greengrass V2 harus dapat menjalankan perintah dengan sudo tanpa diminta kata sandi.

Ikuti langkah-langkah ini pada perangkat yang sedang diuji 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 peran pertukaran token khusus

Anda dapat memilih untuk menggunakan peran IAM khusus sebagai peran pertukaran token yang diasumsikan perangkat yang diuji untuk berinteraksi dengan AWS sumber daya. Untuk informasi tentang membuat peran IAM, lihat Membuat peran IAM di Panduan Pengguna IAM.

Anda harus memenuhi persyaratan berikut untuk memungkinkan IDT menggunakan peran IAM kustom Anda. Kami sangat menyarankan agar Anda menambahkan hanya tindakan kebijakan minimum yang diperlukan untuk peran ini.

- File konfigurasi userdata.json harus diperbarui untuk mengatur parameter ke. `GreengrassV2TokenExchangeRole` `true`
- Peran IAM kustom harus dikonfigurasi dengan kebijakan kepercayaan minimum berikut:

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

- Peran IAM kustom harus dikonfigurasi dengan kebijakan izin minimum berikut:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "iot:DescribeCertificate",
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
```

```
            "logs:PutLogEvents",
            "logs:DescribeLogStreams",
            "iot:Connect",
            "iot:Publish",
            "iot:Subscribe",
            "iot:Receive",
            "iot:ListThingPrincipals",
            "iot:GetThingShadow",
            "iot:UpdateThingShadow",
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:PutObject",
            "s3:AbortMultipartUpload",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":"*"
      }
   ]
}
```

- Nama peran IAM kustom harus cocok dengan sumber daya peran IAM yang Anda tentukan dalam izin IAM untuk pengguna pengujian. Secara default, kebijakan pengguna pengujian mengizinkan akses ke peran IAM yang memiliki `idt-` awalan dalam nama peran mereka. Jika nama peran IAM Anda tidak menggunakan awalan ini, tambahkan `arn:aws:iam::*:role/`*`custom-iam-role-name`* sumber daya ke `roleAliasResources` pernyataan dan `passRoleForResources` pernyataan dalam kebijakan pengguna pengujian Anda, seperti yang ditunjukkan dalam contoh berikut:

  Example **`passRoleForResources`** pernyataan

```
{
   "Sid":"passRoleForResources",
   "Effect":"Allow",
   "Action":"iam:PassRole",
   "Resource":"arn:aws:iam::*:role/custom-iam-role-name",
   "Condition":{
      "StringEquals":{
         "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
         ]
```

```
      }
   }
}
```

Example **`roleAliasResources`** pernyataan

```
{
   "Sid":"roleAliasResources",
   "Effect":"Allow",
   "Action":[
      "iot:CreateRoleAlias",
      "iot:DescribeRoleAlias",
      "iot:DeleteRoleAlias",
      "iot:TagResource",
      "iam:GetRole"
   ],
   "Resource":[
      "arn:aws:iot:*:*:rolealias/idt-*",
      "arn:aws:iam::*:role/custom-iam-role-name"
   ]
}
```

Konfigurasikan perangkat Anda untuk menguji fitur opsional

Bagian ini menjelaskan persyaratan perangkat untuk menjalankan tes IDT untuk fitur Docker dan machine learning (ML) opsional. Fitur ML hanya didukung di IDT v4.9.3. Anda harus memastikan perangkat Anda memenuhi persyaratan ini hanya jika Anda ingin menguji fitur ini. Jika tidak, lanjutkan ke the section called “Konfigurasikan pengaturan IDT”.

Topik

- Persyaratan kualifikasi Docker
- Persyaratan kualifikasi ML
- Persyaratan kualifikasi HSM

Persyaratan kualifikasi Docker

IDT untuk AWS IoT Greengrass V2 menyediakan tes kualifikasi Docker untuk memvalidasi bahwa perangkat Anda dapat menggunakan komponen manajer aplikasi Docker AWS yang disediakan untuk mengunduh gambar kontainer Docker yang dapat Anda jalankan menggunakan komponen

kontainer Docker khusus. Untuk informasi selengkapnya tentang cara membuat konten dokumen kustom, lihat Jalankan kontainer Docker.

Untuk menjalankan tes kualifikasi Docker, perangkat Anda yang sedang diuji harus memenuhi persyaratan berikut untuk men-deploy komponen manajer aplikasi Docker.

- Docker Engine 1.9.1 atau yang lebih baru diinstal pada perangkat inti Greengrass. Versi 20.10 adalah versi terbaru yang diverifikasi untuk bekerja dengan perangkat lunak AWS IoT Greengrass Core. Anda harus menginstal Docker langsung pada perangkat inti sebelum Anda menyebarkan komponen yang menjalankan kontainer Docker.
- Daemon Docker dimulai dan berjalan pada perangkat inti sebelum Anda men-deploy komponen ini.
- Pengguna sistem yang menjalankan komponen kontainer Docker harus memiliki izin root atau administrator, atau Anda harus mengonfigurasi Docker untuk menjalankannya sebagai pengguna non-root atau non-admistrator.
  - Pada perangkat Linux, Anda dapat menambahkan pengguna ke `docker` grup untuk memanggil `docker` perintah tanpa`sudo`.
  - Pada perangkat Windows, Anda dapat menambahkan pengguna ke `docker-users` grup untuk memanggil `docker` perintah tanpa hak istimewa adminstrator.

  Linux or Unix

  Untuk menambahkan`ggc_user`, atau pengguna non-root yang Anda gunakan untuk menjalankan komponen kontainer Docker, ke `docker` grup, jalankan perintah berikut.

  ```
  sudo usermod -aG docker ggc_user
  ```

  Untuk informasi selengkapnya, lihat Mengelola Docker sebagai pengguna non-root.

  Windows Command Prompt (CMD)

  Untuk menambahkan`ggc_user`, atau pengguna yang Anda gunakan untuk menjalankan komponen kontainer Docker, ke `docker-users` grup, jalankan perintah berikut sebagai administrator.

  ```
  net localgroup docker-users ggc_user /add
  ```

Windows PowerShell

Untuk menambahkan`ggc_user`, atau pengguna yang Anda gunakan untuk menjalankan komponen kontainer Docker, ke `docker-users` grup, jalankan perintah berikut sebagai administrator.

```
Add-LocalGroupMember -Group docker-users -Member ggc_user
```

Persyaratan kualifikasi ML

> **Note**
>
> Fitur pembelajaran mesin hanya didukung di IDT v4.9.3.

IDT for AWS IoT Greengrass V2 menyediakan pengujian kualifikasi ML untuk memvalidasi bahwa perangkat Anda dapat menggunakan komponen machine learning yang AWS disediakan untuk melakukan inferensi ML secara lokal menggunakan kerangka kerja Deep Learning Runtime atau Lite ML. TensorFlow Untuk informasi lebih lanjut tentang cara menjalankan inferensi ML pada perangkat Greengrass, lihat Lakukan inferensi machine learning.

Untuk menjalankan tes kualifikasi ML, perangkat Anda yang sedang diuji harus memenuhi persyaratan berikut untuk men-deploy komponen machine learning.

- Pada perangkat inti Greengrass yang menjalankan Amazon Linux 2 atau Ubuntu 18.04, Pustaka GNU C (glibc) versi 2.27 atau yang lebih baru diinstal pada perangkat tersebut.
- Pada perangkat ARMv7L, seperti Raspberry Pi, dependensi untuk OpenCV-Python diinstal pada perangkat. Jalankan perintah berikut untuk menginstal dependensi.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- Perangkat Raspberry Pi yang menjalankan Raspberry Pi OS Bullseye harus memenuhi persyaratan berikut:
  - NumPy 1.22.4 atau yang lebih baru diinstal pada perangkat. Raspberry Pi OS Bullseye menyertakan versi sebelumnya NumPy, sehingga Anda dapat menjalankan perintah berikut untuk meningkatkan NumPy pada perangkat.

```
pip3 install --upgrade numpy
```

- Tumpukan kamera lama diaktifkan di perangkat. Raspberry Pi OS Bullseye menyertakan tumpukan kamera baru yang diaktifkan secara default dan tidak kompatibel, jadi Anda harus mengaktifkan tumpukan kamera lama.

  Untuk mengaktifkan tumpukan kamera lama

  1. Jalankan perintah berikut untuk membuka alat konfigurasi Raspberry Pi.

     ```
     sudo raspi-config
     ```

  2. Pilih Opsi Antarmuka.
  3. Pilih Kamera lama untuk mengaktifkan tumpukan kamera lama.
  4. Reboot Raspberry Pi.

Persyaratan kualifikasi HSM

AWS IoT Greengrass menyediakan komponen penyedia PKCS #11 untuk diintegrasikan dengan Modul Keamanan Perangkat Keras PKCS (HSM) pada perangkat. Pengaturan HSM tergantung pada perangkat Anda dan modul HSM yang telah Anda pilih. Selama konfigurasi HSM yang diharapkan, seperti yang didokumentasikan dalam pengaturan konfigurasi IDT, disediakan, IDT akan memiliki informasi yang diperlukan untuk menjalankan tes kualifikasi fitur opsional ini.

Konfigurasikan pengaturan IDT untuk menjalankan rangkaian AWS IoT Greengrass kualifikasi

Sebelum menjalankan pengujian, Anda harus mengonfigurasi pengaturan untuk AWS kredensil dan perangkat di komputer host Anda.

Konfigurasikan AWS kredensil di config.json

Anda harus mengonfigurasikan kredensial pengguna IAM Anda di file *<device_tester_extract_location>*`/configs/config.json`. Gunakan kredensil untuk IDT untuk pengguna AWS IoT Greengrass V2 yang dibuat di. the section called “Buat dan konfigurasikan Akun AWS” Anda dapat menentukan kredensial Anda dengan salah satu dari dua cara berikut:

- Di file kredensial

- Sebagai variabel lingkungan

Konfigurasikan AWS kredensil 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 AWS kredensi Anda ke `credentials` file 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 V2 untuk menggunakan AWS kredensil dari `credentials` file Anda, edit `config.json` file Anda sebagai berikut:

```
{
  "awsRegion": "region",
  "auth": {
    "method": "file",
    "credentials": {
      "profile": "default"
    }
  }
}
```

> **Note**
>
> Jika Anda tidak menggunakan `default` AWS profil, pastikan untuk mengubah nama profil di `config.json` file Anda. Untuk informasi selengkapnya, lihat Profil Bernama.

Konfigurasikan AWS kredensil dengan variabel lingkungan

Variabel lingkungan adalah variabel yang dikelola oleh sistem operasi dan digunakan oleh perintah sistem. Variabel ini tidak tersimpan jika Anda menutup sesi SSH. IDT untuk AWS IoT Greengrass V2 dapat menggunakan variabel `AWS_ACCESS_KEY_ID` dan `AWS_SECRET_ACCESS_KEY` lingkungan untuk menyimpan AWS kredensil Anda.

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": "region",
  "auth": {
    "method": "environment"
  }
}
```

Konfigurasikan device.json

> **Note**
>
> IDT v4.9.3 mendukung pengujian`ml`,, `docker` dan fitur. `streamManagement` IDT v4.9.4 dan versi yang lebih baru mendukung pengujian. `docker` Jika Anda tidak ingin menguji fitur ini, atur nilai yang sesuai ke no.

Selain AWS kredensil, IDT untuk AWS IoT Greengrass V2 membutuhkan informasi tentang perangkat tempat pengujian dijalankan. Contoh informasi antara lain alamat IP, informasi login, sistem operasi, dan arsitektur CPU.

Anda harus memberikan informasi ini dengan menggunakan templat `device.json` yang terletak di *<device_tester_extract_location>*`/configs/device.json`:

IDT v4.9.3

```
[
  {
    "id": "<pool-id>",
    "sku": "<sku>",
    "features": [
      {
        "name": "arch",
        "value": "x86_64 | armv6l | armv7l | aarch64"
      },
      {
        "name": "ml",
        "value": "dlr | tensorflowlite | dlr,tensorflowlite | no"
      },
      {
        "name": "docker",
        "value": "yes | no"
      },
      {
        "name": "streamManagement",
        "value": "yes | no"
      },
      {
        "name": "hsi",
        "value": "hsm | no"
      }
    ],
    "devices": [
      {
        "id": "<device-id>",
        "operatingSystem": "Linux | Windows",
        "connectivity": {
          "protocol": "ssh",
          "ip": "<ip-address>",
          "port": 22,
          "publicKeyPath": "<public-key-path>",
          "auth": {
            "method": "pki | password",
            "credentials": {
```

```
                "user": "<user-name>",
                "privKeyPath": "/path/to/private/key",
                "password": "<password>"
              }
            }
          }
        }
      ]
    }
]
```

> **Note**
>
> Tentukan `privKeyPath` hanya jika `method` diatur ke `pki`.
> Tentukan `password` hanya jika `method` diatur ke `password`.

Semua properti yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`id`

ID alfanumerik yang ditetapkan pengguna yang secara unik mengidentifikasi kumpulan perangkat 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 yang berkualitas.

> **Note**
>
> Jika Anda ingin mencantumkan perangkat Anda di Katalog AWS Partner Perangkat, SKU yang Anda tentukan di sini harus cocok dengan SKU yang Anda gunakan dalam proses daftar.

`features`

Rangkaian yang berisi fitur perangkat yang didukung. Semua fitur bersifat wajib.

`arch`

Arsitektur sistem operasi yang didukung yang divalidasi oleh oleh tes yang dijalankan. Nilai yang valid adalah:

- `x86_64`
- `armv6l`
- `armv7l`
- `aarch64`

`ml`

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk menggunakan komponen pembelajaran mesin (AWS ML) yang disediakan.

Mengaktifkan fitur ini juga memvalidasi bahwa perangkat dapat melakukan inferensi ML menggunakan kerangka kerja Deep Learning Runtime dan TensorFlow Lite ML.

Nilai yang valid adalah kombinasi dari `dlr` dan `tensorflowlite`, atau no.

`docker`

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk menggunakan komponen Docker application manager () AWS-provided. `aws.greengrass.DockerApplicationManager`

Mengaktifkan fitur ini juga memvalidasi bahwa perangkat dapat mengunduh gambar kontainer Docker dari Amazon ECR.

Nilai yang valid adalah kombinasi dari `yes` atau no.

`streamManagement`

Memvalidasi bahwa perangkat tersebut dapat mengunduh, menginstal, dan menjalankan perintah manajer pengaliran AWS IoT Greengrass.

Nilai yang valid adalah kombinasi dari `yes` atau no.

`hsi`

Memvalidasi bahwa perangkat dapat mengautentikasi koneksi ke AWS IoT dan AWS IoT Greengrass layanan menggunakan kunci pribadi dan sertifikat yang disimpan dalam modul

keamanan perangkat keras (HSM). Pengujian ini juga memverifikasi bahwa komponen penyedia PKCS #11 AWS yang disediakan dapat berinteraksi dengan HSM menggunakan pustaka PKCS #11 yang disediakan vendor. Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras.

Nilai yang valid adalah `hsm` atau no.

> **Note**
>
> Pengujian hanya `hsi` tersedia dengan IDT v4.9.3 dan versi yang lebih baru.

`devices.id`

Pengenal unik yang ditetapkan pengguna untuk perangkat Anda.

`devices.operatingSystem`

Sistem operasi perangkat. Nilai yang didukung adalah `Linux` dan `Windows`.

`connectivity.protocol`

Protokol komunikasi yang digunakan untuk berkomunikasi dengan perangkat ini. Saat ini, satu-satunya nilai yang didukung adalah `ssh` untuk perangkat fisik.

`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.publicKeyPath`

Tidak wajib. Jalur lengkap ke kunci publik digunakan untuk mengautentikasi koneksi ke perangkat yang sedang diuji.

Saat Anda menentukan`publicKeyPath`, IDT memvalidasi kunci publik perangkat saat membuat koneksi SSH ke perangkat yang sedang diuji. Jika nilai ini tidak ditentukan, IDT membuat koneksi SSH, tetapi tidak memvalidasi kunci publik perangkat.

Kami sangat menyarankan Anda menentukan jalur ke kunci publik, dan Anda menggunakan metode aman untuk mengambil kunci publik ini. Untuk klien SSH berbasis baris perintah standar, kunci publik disediakan dalam file. `known_hosts` Jika Anda menentukan file kunci publik terpisah, file ini harus menggunakan format yang sama dengan `known_hosts` file, yaitu, *`ip-address key-type public-key`*. Jika ada beberapa entri dengan alamat ip yang sama, entri untuk key-type yang digunakan oleh IDT harus sebelum entri lain dalam file.

`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.

IDT v4.9.4

```
[
  {
    "id": "<pool-id>",
    "sku": "<sku>",
    "features": [
      {
        "name": "arch",
        "value": "x86_64 | armv6l | armv7l | aarch64"
      },
      {
        "name": "docker",
        "value": "yes | no"
      },
      {
        "name": "hsi",
        "value": "hsm | no"
      }
    ],
    "devices": [
      {
        "id": "<device-id>",
        "operatingSystem": "Linux | Windows",
        "connectivity": {
          "protocol": "ssh",
          "ip": "<ip-address>",
          "port": 22,
          "publicKeyPath": "<public-key-path>",
          "auth": {
            "method": "pki | password",
            "credentials": {
              "user": "<user-name>",
              "privKeyPath": "/path/to/private/key",
              "password": "<password>"
            }
          }
        }
      }
```

```
    ]
  }
]
```

> **Note**
>
> Tentukan `privKeyPath` hanya jika `method` diatur ke `pki`.
> Tentukan `password` hanya jika `method` diatur ke `password`.

Semua properti yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`id`

ID alfanumerik yang ditetapkan pengguna yang secara unik mengidentifikasi kumpulan perangkat 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 yang berkualitas.

> **Note**
>
> Jika Anda ingin mencantumkan perangkat Anda di Katalog AWS Partner Perangkat, SKU yang Anda tentukan di sini harus cocok dengan SKU yang Anda gunakan dalam proses daftar.

`features`

Rangkaian yang berisi fitur perangkat yang didukung. Semua fitur bersifat wajib.

`arch`

Arsitektur sistem operasi yang didukung yang divalidasi oleh oleh tes yang dijalankan. Nilai yang valid adalah:

- `x86_64`

- `armv6l`
- `armv7l`
- `aarch64`

`docker`

Memvalidasi bahwa perangkat memenuhi semua dependensi teknis yang diperlukan untuk menggunakan komponen Docker application manager () AWS-provided. `aws.greengrass.DockerApplicationManager`

Mengaktifkan fitur ini juga memvalidasi bahwa perangkat dapat mengunduh gambar kontainer Docker dari Amazon ECR.

Nilai yang valid adalah kombinasi dari `yes` atau `no`.

`hsi`

Memvalidasi bahwa perangkat dapat mengautentikasi koneksi ke AWS IoT dan AWS IoT Greengrass layanan menggunakan kunci pribadi dan sertifikat yang disimpan dalam modul keamanan perangkat keras (HSM). Pengujian ini juga memverifikasi bahwa komponen penyedia PKCS #11 AWS yang disediakan dapat berinteraksi dengan HSM menggunakan pustaka PKCS #11 yang disediakan vendor. Untuk informasi selengkapnya, lihat Integrasi keamanan perangkat keras.

Nilai yang valid adalah `hsm` atau `no`.

> **Note**
>
> Pengujian hanya `hsi` tersedia dengan IDT v4.9.3 dan versi yang lebih baru.

`devices.id`

Pengenal unik yang ditetapkan pengguna untuk perangkat Anda.

`devices.operatingSystem`

Sistem operasi perangkat. Nilai yang didukung adalah `Linux` dan `Windows`.

`connectivity.protocol`

Protokol komunikasi yang digunakan untuk berkomunikasi dengan perangkat ini. Saat ini, satu-satunya nilai yang didukung adalah `ssh` untuk perangkat fisik.

`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.publicKeyPath`

Tidak wajib. Jalur lengkap ke kunci publik digunakan untuk mengautentikasi koneksi ke perangkat yang sedang diuji.

Saat Anda menentukan`publicKeyPath`, IDT memvalidasi kunci publik perangkat saat membuat koneksi SSH ke perangkat yang sedang diuji. Jika nilai ini tidak ditentukan, IDT membuat koneksi SSH, tetapi tidak memvalidasi kunci publik perangkat.

Kami sangat menyarankan Anda menentukan jalur ke kunci publik, dan Anda menggunakan metode aman untuk mengambil kunci publik ini. Untuk klien SSH berbasis baris perintah standar, kunci publik disediakan dalam file. `known_hosts` Jika Anda menentukan file kunci publik terpisah, file ini harus menggunakan format yang sama dengan `known_hosts` file, yaitu, *`ip-address key-type public-key`*. Jika ada beberapa entri dengan alamat ip yang sama, entri untuk key-type yang digunakan oleh IDT harus sebelum entri lain dalam file.

`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.

Konfigurasikan userdata.json

IDT untuk AWS IoT Greengrass V2 juga membutuhkan informasi tambahan tentang lokasi artefak uji dan AWS IoT Greengrass perangkat lunak.

Anda harus memberikan informasi ini dengan menggunakan templat `userdata.json` yang terletak di *<device_tester_extract_location>*`/configs/userdata.json`:

```
{
    "TempResourcesDirOnDevice": "/path/to/temp/folder",
    "InstallationDirRootOnDevice": "/path/to/installation/folder",
    "GreengrassNucleusZip": "/path/to/aws.greengrass.nucleus.zip",
    "PreInstalled": "yes/no",
    "GreengrassV2TokenExchangeRole": "custom-iam-role-name",
 "hsm": {
        "greengrassPkcsPluginJar": "/path/to/aws.greengrass.crypto.Pkcs11Provider-
latest.jar",
        "pkcs11ProviderLibrary": "/path/to/pkcs11-vendor-library",
        "slotId": "slot-id",
        "slotLabel": "slot-label",
        "slotUserPin": "slot-pin",
        "keyLabel": "key-label",
        "preloadedCertificateArn": "certificate-arn"
```

```
        "rootCA": "path/to/root-ca"
    }
}
```

Semua properti yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`TempResourcesDirOnDevice`

Jalur lengkap ke folder sementara pada perangkat yang diuji untuk menyimpan artefak uji. Pastikan bahwa izin sudo tidak diperlukan untuk menulis ke direktori ini.

> **Note**
>
> IDT menghapus isi folder ini ketika selesai menjalankan tes.

`InstallationDirRootOnDevice`

Jalur penuh ke folder pada perangkat tempat menginstal AWS IoT Greengrass. Untuk PreInstalled Greengrass, ini adalah jalur ke direktori instalasi Greengrass.

Anda harus mengatur izin file yang diperlukan untuk folder ini. Jalankan perintah berikut untuk setiap folder di jalur instalasi.

```
sudo chmod 755 folder-name
```

`GreengrassNucleusZip`

Jalur penuh ke file ZIP inti Greengrass (`greengrass-nucleus-latest.zip`) pada komputer host Anda. Bidang ini tidak diperlukan untuk pengujian dengan PreInstalled Greengrass.

> **Note**
>
> Untuk informasi tentang versi inti Greengrass yang didukung untuk IDT, lihat. AWS IoT GreengrassVersi IDT terbaru untuk V2 AWS IoT Greengrass Untuk mengunduh perangkat lunak Greengrass terbaru, lihat Mengunduh perangkat lunak. AWS IoT Greengrass

`PreInstalled`

Fitur ini tersedia untuk IDT v4.5.8 dan versi yang lebih baru hanya pada perangkat Linux.

(Opsional) Ketika nilainya *ya*, IDT akan menganggap jalur yang disebutkan sebagai direktori tempat Greengrass diinstal. `InstallationDirRootOnDevice`

Untuk informasi selengkapnya tentang cara menginstal Greengrass di perangkat Anda, lihat. Instal perangkat lunak inti AWS IoT Greengrass dengan penyediaan sumber daya otomatis Jika menginstal dengan penyediaan manual, sertakan langkah "Tambahkan AWS IoT benda ke grup hal baru atau yang sudah ada" saat membuat AWS IoT sesuatu secara manual. IDT mengasumsikan bahwa grup benda dan benda dibuat selama pengaturan instalasi. Pastikan bahwa nilai-nilai ini tercermin dalam `effectiveConfig.yaml` file. IDT memeriksa file di `effectiveConfig.yaml` bawah`<InstallationDirRootOnDevice>/config/effectiveConfig.yaml`.

Untuk menjalankan tes dengan HSM, pastikan bahwa `aws.greengrass.crypto.Pkcs11Provider` bidang diperbarui die`ffectiveConfig.yaml`.

`GreengrassV2TokenExchangeRole`

(Opsional) Peran IAM kustom yang ingin Anda gunakan sebagai peran pertukaran token yang diasumsikan perangkat yang diuji untuk berinteraksi dengan AWS sumber daya.

> **Note**
>
> IDT menggunakan peran IAM khusus ini alih-alih membuat peran pertukaran token default selama pengujian dijalankan. Jika Anda menggunakan peran kustom, Anda dapat memperbarui izin IAM untuk pengguna pengujian untuk mengecualikan `iamResourcesUpdate` pernyataan yang memungkinkan pengguna membuat dan menghapus peran dan kebijakan IAM.

Untuk informasi selengkapnya tentang membuat peran IAM kustom sebagai peran pertukaran token Anda, lihatKonfigurasikan peran pertukaran token khusus.

`hsm`

Fitur ini tersedia untuk IDT v4.5.1 dan yang lebih baru.

(Opsional) Informasi konfigurasi untuk pengujian dengan Modul Keamanan AWS IoT Greengrass Perangkat Keras (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`.

> **Warning**
>
> Konfigurasi HSM dapat dianggap sebagai data sensitif jika modul keamanan perangkat keras dibagi antara IDT dan sistem lain. Dalam situasi ini, Anda dapat menghindari mengamankan nilai konfigurasi ini dalam teks biasa dengan menyimpannya dalam AWS parameter SecureString Parameter Store dan mengonfigurasi IDT untuk mengambilnya selama eksekusi pengujian. Untuk informasi selengkapnya, lihat ???

`hsm.greengrassPkcsPluginJar`

Jalur lengkap ke komponen penyedia PKCS #11 yang Anda unduh ke mesin host IDT. AWS IoT Greengrass menyediakan komponen ini sebagai file JAR yang dapat Anda unduh untuk ditentukan sebagai plugin penyediaan selama instalasi. Anda dapat mengunduh versi terbaru dari file JAR komponen sebagai URL berikut: https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/aws.greengrass.crypto.Pkcs11Provider-latest.jar.

`hsm.pkcs11ProviderLibrary`

Jalur lengkap ke pustaka PKCS #11 yang disediakan oleh vendor modul keamanan perangkat keras (HSM) untuk berinteraksi dengan HSM.

`hsm.slotId`

ID slot yang digunakan untuk mengidentifikasi slot HSM tempat Anda memuat kunci dan sertifikat.

`hsm.slotLabel`

Label slot yang digunakan untuk mengidentifikasi slot HSM tempat Anda memuat kunci dan sertifikat.

`hsm.slotUserPin`

PIN pengguna yang digunakan IDT untuk mengautentikasi perangkat lunak AWS IoT Greengrass Core ke HSM.

> **Note**
>
> Sebagai praktik keamanan terbaik, jangan gunakan PIN pengguna yang sama pada perangkat produksi.

`hsm.keyLabel`

Label yang digunakan untuk mengidentifikasi kunci dalam modul perangkat keras. Kunci dan sertifikat harus menggunakan label kunci yang sama.

`hsm.preloadedCertificateArn`

Nama Sumber Daya Amazon (ARN) dari sertifikat perangkat yang diunggah di cloud. AWS IoT

Anda sebelumnya harus membuat sertifikat ini menggunakan kunci di HSM, mengimpornya ke HSM Anda, dan mengunggahnya ke cloud. AWS IoT Untuk informasi tentang membuat dan mengimpor sertifikat, lihat dokumentasi untuk HSM Anda.

Anda harus mengunggah sertifikat ke akun dan Wilayah yang sama yang Anda berikan di config.json. . Untuk informasi selengkapnya tentang mengunggah sertifikat AWS IoT, lihat Mendaftarkan sertifikat klien secara manual di Panduan AWS IoT Pengembang.

`hsm.rootCAPath`

(Opsional) Jalur lengkap pada mesin host IDT ke otoritas sertifikat root (CA) yang menandatangani sertifikat Anda. Ini diperlukan jika sertifikat di HSM Anda yang dibuat tidak ditandatangani oleh Amazon root CA.

Ambil konfigurasi dari AWS Parameter Store

AWS IoT Device Tester (IDT) menyertakan fitur opsional untuk mengambil nilai konfigurasi dari AWS Systems Manager Parameter Store. AWS Parameter Store memungkinkan penyimpanan konfigurasi yang aman dan terenkripsi. Saat dikonfigurasi, IDT dapat mengambil parameter dari AWS Parameter Store sebagai pengganti menyimpan parameter dalam teks biasa di dalam file. `userdata.json` Ini berguna untuk semua data sensitif yang harus disimpan terenkripsi, seperti: kata sandi, pin, dan rahasia lainnya.

1. Untuk menggunakan fitur ini, Anda harus memperbarui izin yang digunakan dalam membuat pengguna IDT Anda untuk mengizinkan GetParameter tindakan pada parameter yang IDT dikonfigurasi untuk digunakan. Di bawah ini adalah contoh pernyataan izin yang dapat ditambahkan ke pengguna IDT. Untuk informasi selengkapnya, lihat AWS Systems Manager panduan pengguna.

   ```
   {
          "Sid":"parameterStoreResources",
   ```

```
        "Effect": "Allow",
         "Action": [
             "ssm:GetParameter"
         ],
         "Resource": "arn:aws:ssm:*:*:parameter/IDT*"
}
```

Izin di atas dikonfigurasi untuk memungkinkan pengambilan semua parameter dengan nama yang dimulai dengan`IDT`, dengan menggunakan karakter wildcard. * Anda harus menyesuaikan ini dengan kebutuhan Anda sehingga IDT memiliki akses untuk mengambil parameter yang dikonfigurasi berdasarkan penamaan parameter yang Anda gunakan.

2. Anda perlu menyimpan nilai konfigurasi Anda di dalam AWS Paramater Store. Ini dapat dilakukan dari AWS konsol atau dari AWS CLI. AWS Parameter Store memungkinkan Anda memilih penyimpanan terenkripsi atau tidak terenkripsi. Untuk penyimpanan nilai sensitif seperti rahasia, kata sandi, dan pin, Anda harus menggunakan opsi terenkripsi yang merupakan jenis parameter. SecureString Untuk mengunggah parameter menggunakan AWS CLI, Anda dapat menggunakan perintah berikut:

   ```
   aws ssm put-parameter --name IDT-example-name --value IDT-example-value --type
    SecureString
   ```

   Anda dapat memverifikasi bahwa parameter disimpan menggunakan perintah berikut. (Opsional) Gunakan `--with-decryption` bendera untuk mengambil parameter yang didekripsiSecureString .

   ```
   aws ssm get-parameter --name IDT-example-name
   ```

   Menggunakan AWS CLI akan mengunggah parameter di AWS wilayah pengguna CLI saat ini dan IDT akan mengambil parameter dari wilayah yang dikonfigurasi. `config.json` Untuk memeriksa wilayah Anda dari AWS CLI, gunakan yang berikut ini:

   ```
   aws configure get region
   ```

3. Setelah Anda memiliki nilai konfigurasi di AWS Cloud, Anda dapat memperbarui nilai apa pun di dalam konfigurasi IDT untuk diambil dari Cloud. AWS Untuk melakukannya, Anda menggunakan placeholder dalam konfigurasi formulir IDT Anda `{{AWS.Parameter.parameter_name}}` untuk mengambil parameter dengan nama itu dari Parameter Store. AWS

Misalnya, Anda ingin menggunakan `IDT-example-name` parameter dari Langkah 2 sebagai KeyLabel HSM dalam konfigurasi HSM Anda. Untuk melakukan ini, Anda dapat memperbarui `userdata.json` sebagai berikut:

```
"hsm": {
        "keyLabel": "{{AWS.Parameter.IDT-example-name}}",
        [...]
    }
```

IDT akan mengambil nilai parameter ini saat runtime yang disetel ke Langkah 2`IDT-example-value`. Konfigurasi ini mirip dengan pengaturan `"keyLabel": "IDT-example-value"` tetapi, sebaliknya, nilai itu disimpan sebagai terenkripsi di AWS Cloud.

Jalankan rangkaian kualifikasi AWS IoT Greengrass

Setelah Anda mengatur konfigurasi yang diperlukan, Anda bisa memulai tes. Waktu aktif dari rangkaian uji penuh tergantung pada perangkat keras Anda. Sebagai referensi, dibutuhkan sekitar 30 menit untuk menyelesaikan rangkaian tes penuh pada Raspberry Pi 3B.

Gunakan `run-suite` berikut untuk menjalankan rangkaian tes.

```
devicetester_[linux | mac | win]_x86-64 run-suite  \\
    --suite-id suite-id  \\
    --group-id group-id  \\
    --pool-id your-device-pool \\
    --test-id test-id  \\
    --update-idt y|n  \\
    --userdata userdata.json
```

Semua opsi adalah opsional. Misalnya, Anda dapat menghilangkan `pool-id` jika anda hanya mempunyai satu kolam perangkat, yang merupakan serangkaian perangkat yang sama, yang ditentukan dalam file `device.json` Anda. Atau, Anda bisa menghilangkan `suite-id` jika Anda ingin menjalankan versi rangkaian uji terbaru di folder `tests`.

> **Note**
>
> IDT meminta Anda jika versi rangkaian tes yang lebih baru tersedia secara online. Untuk informasi selengkapnya, lihat the section called “Versi rangkaian tes”.

Contoh perintah untuk menjalankan rangkaian kualifikasi

Contoh baris perintah berikut menunjukkan cara menjalankan tes kualifikasi untuk kolam perangkat. Untuk informasi selengkapnya tentang `run-suite` dan perintah IDT CLI lainnya, lihat the section called “Perintah IDT”.

Gunakan perintah berikut untuk menjalankan semua grup uji dalam rangkaian tes tertentu. Perintah `list-suites` mencantumkan rangkaian uji yang ada di folder `tests`.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
    --suite-id GGV2Q_1.0.0 \
    --pool-id <pool-id> \
    --userdata userdata.json
```

Gunakan perintah berikut untuk menjalankan semua grup uji tertentu dalam rangkaian tes. Perintah `list-groups` mencantumkan grup uji dalam rangkaian uji.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
    --suite-id GGV2Q_1.0.0 \
    --group-id <group-id> \
    --pool-id <pool-id> \
    --userdata userdata.json
```

Gunakan perintah berikut untuk menjalankan uji kasus tertentu dalam grup uji.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
    --group-id <group-id> \
    --test-id <test-id> \
    --userdata userdata.json
```

Gunakan perintah berikut untuk menjalankan beberapa uji kasus dalam grup uji.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
    --group-id <group-id> \
    --test-id <test-id1>,<test-id2>
    --userdata userdata.json
```

Gunakan perintah berikut untuk mencantumkan semua uji kasus dalam grup uji.

```
devicetester_[linux | mac | win]_x86-64 list-test-cases --group-id <group-id>
```

Kami menyarankan Anda menjalankan rangkaian pengujian kualifikasi lengkap, yang menjalankan dependensi grup pengujian dalam urutan yang benar. Jika Anda memilih untuk menjalankan grup pengujian tertentu, sebaiknya jalankan grup uji pemeriksa dependensi terlebih dahulu untuk memastikan semua dependensi Greengrass diinstal sebelum menjalankan grup pengujian terkait. Sebagai contoh:

- Jalankan `coredependencies` sebelum menjalankan grup uji kualifikasi inti.

IDT untuk perintah V2 AWS IoT Greengrass

Perintah IDT terletak di direktori *<device-tester-extract-location>*`/bin`. Untuk menjalankan rangkaian tes, Anda memberikan perintah dalam format berikut:

`help`

Mencantumkan 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 menjalankan semua grup pengujian yang sesuai dalam rangkaian pengujian

tergantung pada `device.json` pengaturan yang dikonfigurasi. IDT tidak menjalankan grup pengujian apa pun yang tidak didukung perangkat berdasarkan pengaturan yang dikonfigurasi, meskipun grup pengujian tersebut ditentukan dalam `group-id` daftar.

- `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 suatu kolam jika Anda memiliki beberapa kolam perangkat yang ditentukan dalam file `device.json` Anda.
- `stop-on-first-failure`. Mengonfigurasi IDT untuk berhenti berjalan pada kegagalan pertama. Gunakan opsi ini dengan `group-id` ketika Anda ingin men-debug grup uji tertentu. Jangan gunakan opsi ini saat menjalankan rangkaian uji penuh untuk menghasilkan laporan kualifikasi.
- `update-idt`. Menetapkan respons bagi prompt untuk memperbarui IDT. Respons `Y` menghentikan pelaksanaan tes jika IDT mendeteksi ada versi yang lebih baru. Respons `N` melanjutkan pelaksanaan tes.
- `userdata`. Path lengkap ke file `userdata.json` yang berisi informasi tentang path artefak uji. Opsi ini diperlukan untuk perintah `run-suite`. File `userdata.json` harus terletak di direktori *`devicetester_extract_location`*/devicetester_ggv2_*`[win|mac|linux]`*/configs/.

Untuk informasi lebih lanjut tentang opsi `run-suite`, gunakan opsi `help`:

```
devicetester_[linux | mac | win]_x86-64 run-suite -h
```

Memahami hasil dan log

Bagian ini menjelaskan cara melihat dan menafsirkan laporan hasil IDT dan log.

Untuk memecahkan masalah kesalahan, lihatPenyelesaian masalah IDT untuk V2 AWS IoT Greengrass.

Melihat Hasil

Saat berjalan, IDT menuliskan kesalahan ke konsol, file log, dan laporan tes. Setelah IDT menyelesaikan rangkaian tes kualifikasi, ia akan menghasilkan dua laporan uji. Laporan-laporan ini terletak di *`<device-tester-extract-location>`*`/results/`*`<execution-id>`*`/`. Kedua laporan tersebut menangkap hasil dari menjalankan rangkaian uji kualifikasi.

`awsiotdevicetester_report.xml` adalah laporan uji kualifikasi yang Anda kirimkan ke AWS untuk mencantumkan perangkat Anda di Katalog Perangkat AWS Partner. 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 pustakan yang diuji berdasarkan fitur perangkat, seperti akses sumber daya lokal, bayangan, dan MQTT.

Laporan `GGV2Q_Result.xml` 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.
- Perincian hasil uji menurut fungsionalitas AWS IoT Greengrass yang diuji.

Menafsirkan hasil AWS IoT Device Tester

Bagian laporan di `awsiotdevicetester_report.xml` atau `awsiotdevicetester_report.xml` mencantumkan tes yang dijalankan dan hasilnya.

Tag XML pertama `<testsuites>` berisi ringkasan tes yang dijalankan. Misalnya:

```
<testsuites name="GGQ 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 diperlukan untuk menjalankan rangkaian kualifikasi.

`tests`

Jumlah tes yang dijalankan.

`failures`

Jumlah tes yang dijalankan, tetapi tidak lulus.

`errors`

Jumlah tes yang tidak bisa dijalankan oleh IDT.

`disabled`

Abaikan atribut ini. Atribut ini tidak digunakan.

File `awsiotdevicetester_report.xml` berisi sebuah 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` wajib mengirimkan forum Anda untuk kualifikasi. Potongan berikut menunjukkan bagaimana informasi ini muncul di file `awsiotdevicetester_report.xml`.

```
<name="aws-iot-greengrass-v2-core" value="supported" type="required"></feature>
```

Jika tidak terdapat kegagalan pengujian atau kesalahan untuk fitur yang diperlukan, perangkat Anda memenuhi persyaratan teknis untuk menjalankan AWS IoT Greengrass dan dapat bekerja sama dengan layanan AWS IoT. 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 pengujian yang dijalankan 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 menjalankan uji kasus.

Ketika tes gagal atau kesalahan terjadi, tanda `<failure>` atau `<error>` akan ditambahkan ke tanda `<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 tes yang berjalan di *`<devicetester-extract-location>`*`/results/`*`<execution-id>`*`/logs`. Dua rangkaian log yang dihasilkan:

`test_manager.log`

Log yang dihasilkan dari komponen Test Manager AWS IoT Device Tester (misalnya, log yang terkait dengan konfigurasi, pengurutan tes, dan pembuatan laporan).

`<test-case-id>``.log (for example, lambdaDeploymentTest.log)`

Log dari kasus uji dalam grup uji, termasuk log dari perangkat yang diuji. Dimulai dengan IDT v4.2.0, IDT mengelompokkan log pengujian untuk setiap kasus pengujian dalam folder *`test-case-id<>`* terpisah di dalam*`<devicetester-extract-location>`*`/results/`*`<execution-id>`*`/logs/`*`<test-group-id>/`* direktori.

Gunakan IDT untuk mengembangkan dan menjalankan rangkain tes Anda sendiri

Dimulai di IDT v4.0.1, IDT untuk V2 AWS IoT Greengrass menggabungkan pengaturan konfigurasi standar dan format hasil dengan lingkungan rangkaian tes yang memungkinkan Anda untuk mengembangkan rangkaian tes kustom untuk perangkat Anda dan perangkat lunak perangkat. Anda dapat menambahkan tes 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 suite uji khusus

- 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 suite pengujian 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 AWS IoT Device Tester untuk AWS IoT Greengrass

Unduh versi terbaru IDT dan ekstrak perangkat lunak ke lokasi (*< device-tester-extract-location >*) pada sistem file Anda di mana Anda memiliki izin baca/tulis.

> **Note**
>
> IDT tidak mendukung untuk 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 yang menyediakan IDT dengan informasi tentang cara menjalankan test suite.
- 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 untuk rangkaian pengujian 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 dan jalankan sampel rangkaian tes IDT.

Untuk memulai membuat rangkaian tes kustom di Python, lihat Tutorial: Kembangkan rangkaian tes IDT sederhana.

Tutorial: Bangun dan jalankan sampel rangkaian tes IDT

Unduhan AWS IoT Device Tester meliputi kode sumber untuk rangkaian tes sampel. Anda dapat menyelesaikan tutorial ini untuk membangun dan menjalankan sampel rangkaian tes untuk memahami bagaimana Anda dapat menggunakan IDT untuk AWS IoT Greengrass untuk menjalankan rangkaian tes kustom.

Dalam tutorial ini, Anda akan melakukan 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 AWS IoT Device Tester terbaru
  - 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`

Opsional. 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 dalam `IDTSampleSuitePython_1.0.0` folder untuk memahami bagaimana rangkaian pengujian sampel terstruktur, dan untuk melihat berbagai contoh executable kasus uji dan file JSON konfigurasi pengujian.

> **Note**
>
> Rangkaian pengujian sampel menyertakan kode sumber python. Jangan sertakan informasi sensitif dalam kode rangkaian pengujian Anda.

Langkah berikutnya: Gunakan IDT untuk menjalankan rangkaian tes sampel yang Anda buat.

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
```

Memecahkan 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

Rangkaian tes menggabungkan hal berikut:

- Executable tes yang berisi logika tes
- File konfigurasi yang menjelaskan rangkaian pengujian

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
3. Buat executable uji kasus

4. Jalankan rangkaian tes

Prasyarat

Untuk menyelesaikan tutorial ini, Anda memerlukan hal berikut ini:

- Persyaratan komputer host
  - Versi AWS IoT Device Tester terbaru
  - 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

Rangkaian pengujian Anda harus berisi file konfigurasi wajib berikut:

File konfigurasi yang diperlukan

`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`

Opsional. 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
```

Memecahkan 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 yang Anda sertakan saat Anda menulis rangkaian pengujian kustom.

File konfigurasi yang diperlukan

`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 konfigurasi opsional

`test_orchestrator.yaml` atau `state_machine.json`

Menentukan bagaimana tes akan dijalankan ketika IDT menjalankan rangkaian tes. SsE. Konfigurasikan test_orchestrator.yaml

> **Note**
>
> Mulai IDT v4.5.1, Anda menggunakan `test_orchestrator.yaml` file untuk menentukan alur kerja pengujian. Di versi IDT sebelumnya, Anda menggunakan `state_machine.json` file tersebut. Untuk informasi tentang mesin negara, lihatKonfigurasikan state machine IDT.

`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 ditempatkan di Anda *<custom-test-suite-folder>* seperti yang ditunjukkan di sini.

```
<custom-test-suite-folder>
### suite
    ### suite.json
    ### test_orchestrator.yaml
```

```
    ### 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`

Opsional. 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 `id` harus cocok dengan nama folder grup pengujian tempat `group.json` file berada, dan tidak dapat 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`

Opsional. 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 `id` harus cocok dengan nama folder kasus uji tempat `test.json` file berada, dan tidak dapat berisi garis bawah (_). ID digunakan dalam laporan pengujian yang dihasilkan ID.

`title`

Nama deskriptif untuk uji kasus. Nama tersebut ditampilkan dalam IDT CLI untuk test runner.

`details`

Deskripsi singkat tentang tujuan dari uji kasus.

`requireDUT`

Opsional. 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`

Opsional. 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`

Opsional. 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`

Opsional. 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`

Opsional. Argumen yang akan disediakan untuk menjalankan executable uji kasus.

`environmentVariables`

Opsional. 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 test_orchestrator.yaml

Orkestrator pengujian adalah konstruksi yang mengontrol alur eksekusi test suite. 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 pengujian Anda tidak menyertakan orkestrator pengujian yang ditentukan pengguna, IDT akan menghasilkan orkestrator pengujian untuk Anda.

Orkestrator uji 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 fungsi orkestrator pengujian IDT, lihat. Konfigurasikan orkestrasi uji 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 `userdata.json` file tersedia dalam konteks IDT dan dalam konteks orkestrator pengujian.

Konfigurasikan orkestrasi uji IDT

Mulai di IDT v4.5.1, IDT termasuk baruorkestrasi ujikomponen. Orkestrator uji adalah komponen IDT yang mengontrol aliran eksekusi test suite, dan menghasilkan laporan uji setelah IDT selesai menjalankan semua tes. Orkestrator uji menentukan pemilihan tes dan urutan pengujian dijalankan berdasarkan aturan yang ditetapkan pengguna.

Jika rangkaian tes Anda tidak menyertakan orkestrasi pengujian yang ditetapkan pengguna, IDT akan membuat orkestrasi pengujian untuk Anda.

Orkestrasi uji 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.

Orkestrator tes menggantikan orkestrator tes IDT. Kami sangat menyarankan Anda menggunakan orkestrator tes untuk mengembangkan suite tes Anda alih-alih orkestrator tes IDT. Orkestrator uji menyediakan fitur yang lebih baik berikut:

- Menggunakan format deklaratif dibandingkan dengan format imperatif bahwa mesin negara IDT menggunakan. Hal ini mengizinkan Anda untukmenentukantes mana yang ingin Anda jalankan danketikaAnda ingin menjalankan mereka.
- Mengelola penanganan kelompok tertentu, pembuatan laporan, penanganan kesalahan, dan pelacakan hasilsehingga Anda tidak diperlukanuntuk mengelola tindakan ini secara manual.
- Menggunakan format YAML, yang mendukung komentar secara default.
- Membutuhkan80 persenruang disk kurang dari orkestrator tes untuk menentukan alur kerja yang sama.
- Menambahkan validasi pra-tes untuk memverifikasi bahwa definisi alur kerja Anda tidak mengandung ID pengujian yang salah atau dependensi melingkar.

Format orkestrator uji

Anda dapat menggunakan templat berikut untuk mengonfigurasi file *<custom-test-suite-folder>*`/suite/test_orchestrator.yaml` Anda:

```
Aliases:
  string: context-expression

ConditionalTests:
  - Condition: context-expression
    Tests:
      - test-descriptor

Order:
  - - group-descriptor
```

```
    - group-descriptor

Features:
  - Name: feature-name
    Value: support-description
    Condition: context-expression
    Tests:
        - test-descriptor
    OneOfTests:
        - test-descriptor
    IsRequired: boolean
```

Semua kolom yang berisi nilai wajib diisi seperti yang dijelaskan di sini:

`Aliases`

Tidak wajib. String yang ditentukan pengguna yang memetakan ekspresi konteks. Alias memungkinkan Anda untuk menghasilkan nama ramahmengidentifikasi ekspresi konteks dalam konfigurasi orkestrator pengujian Anda. Hal ini sangat berguna jika Anda membuat ekspresi konteks kompleks atau ekspresi yang Anda gunakan di beberapa tempat.

Anda dapat menggunakan ekspresi konteks untuk menyimpan kueri konteks yang memungkinkan Anda mengakses data dari konfigurasi IDT lainnya. Untuk informasi selengkapnya, lihat Akses data dalam konteks.

Example Contoh

```
Aliases:
    FizzChosen: "'{{$pool.features[?(@.name == 'Fizz')].value[0]}}' == 'yes'"
    BuzzChosen: "'{{$pool.features[?(@.name == 'Buzz')].value[0]}}' == 'yes'"
    FizzBuzzChosen: "'{{$aliases.FizzChosen}}' && '{{$aliases.BuzzChosen}}'"
```

`ConditionalTests`

Tidak wajib. Daftar kondisi, dan kasus uji yang sesuai yang dijalankan ketika setiap kondisi puas. Setiap kondisi dapat memiliki beberapa kasus uji; Namun, Anda dapat menetapkan kasus uji tertentu untuk hanya satu kondisi.

Secara default, IDT menjalankan setiap kasus uji yang tidak ditetapkan ke kondisi dalam daftar ini. Jika Anda tidak menentukan bagian ini, IDT akan menjalankan semua grup pengujian.

Setiap item dalam`ConditionalTests`mencakup parameter-parameter berikut:

`Condition`

Ekspresi konteks yang mengevaluasi keBooleannilai. Jika nilai yang dievaluasi benar, IDT menjalankan kasus uji yang ditentukan dalam`Tests`parameter.

`Tests`

Daftar deskriptor tes.

Setiap deskriptor pengujian menggunakan ID grup uji dan satu atau beberapa ID kasus uji untuk mengidentifikasi pengujian individual yang dijalankan dari kelompok uji tertentu. Deskriptor pengujian menggunakan format berikut:

```
GroupId: group-id
CaseIds: [test-id, test-id] # optional
```

Example Contoh

Contoh berikut menggunakan ekspresi konteks generik yang dapat Anda definisikan sebagai`Aliases`.

```
ConditionalTests:
    - Condition: "{{$aliases.Condition1}}"
      Tests:
          - GroupId: A
          - GroupId: B
    - Condition: "{{$aliases.Condition2}}"
      Tests:
          - GroupId: D
    - Condition: "{{$aliases.Condition1}} || {{$aliases.Condition2}}"
      Tests:
          - GroupId: C
```

Berdasarkan kondisi yang ditetapkan, IDT memilih kelompok uji sebagai berikut:

- Jika`Condition1`benar, IDT menjalankan tes pada kelompok uji A, B, dan C.
- Jika`Condition2`benar, IDT menjalankan tes pada kelompok uji C dan D.

`Order`

Tidak wajib. Urutan untuk menjalankan tes. Anda menentukan urutan tes di tingkat kelompok uji. Jika Anda tidak menentukan bagian ini, IDT akan menjalankan semua grup pengujian yang berlaku dalam urutan acak. Nilai dari`Order`adalah daftar daftar deskriptor kelompok. Setiap

kelompok uji yang tidak Anda cantumkan`Order`, dapat dijalankan secara parallel dengan kelompok uji lainnya yang terdaftar.

Setiap daftar deskriptor grup berisi salah satu deskriptor grup lainnya, dan mengidentifikasi urutan untuk menjalankan grup yang ditentukan dalam setiap deskriptor. Anda dapat menggunakan format berikut untuk menentukan deskriptor grup individual:

- *`group-id`*—ID grup dari grup uji yang ada.
- `[`*`group-id`*`, `*`group-id`*`]`—Daftar kelompok uji yang dapat dijalankan dalam urutan apapun relatif terhadap satu sama lain.
- `"*"`Wildcard. Ini setara dengan daftar semua kelompok uji yang belum ditentukan dalam daftar deskriptor grup saat ini.

Nilai untuk`Order`juga harus memenuhi persyaratan berikut:

- ID grup uji yang Anda tetapkan dalam deskriptor grup harus ada di rangkaian pengujian Anda.
- Setiap daftar deskriptor grup harus menyertakan setidaknya satu grup uji.
- Setiap daftar deskriptor grup harus berisi ID grup unik. Anda tidak dapat mengulangi ID grup uji dalam deskriptor grup individual.
- Daftar deskriptor grup dapat memiliki paling banyak satu deskriptor wildcard. Deskriptor grup wildcard harus menjadi item pertama atau terakhir dalam daftar.

Example Contoh

Untuk test suite yang berisi kelompok tes A, B, C, D, dan E, daftar berikut contoh menunjukkan cara yang berbeda untuk menentukan bahwa IDT pertama harus menjalankan tes kelompok A, kemudian menjalankan tes kelompok B, dan kemudian menjalankan tes kelompok C, D, dan E dalam urutan apapun.

- ```
  Order:
      - - A
        - B
        - [C, D, E]
  ```
- ```
  Order:
      - - A
        - B
        - "*"
  ```
- ```
  Order:
      - - A
  ```
  ```

```
    - B

  - - B
    - C

  - - B
    - D

  - - B
    - E
```

`Features`

Tidak wajib. Daftar fitur produk yang Anda ingin IDT untuk menambahkan ke`awsiotdevicetester_report.xml`berkas. Jika Anda tidak menentukan bagian ini, IDT tidak akan menambahkan fitur produk apa pun ke laporan.

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. Masuka`wsiotdevicetester_report.xml`, fitur produk ditetapkan sebagai`supported`,`not-supported`, atau nilai yang ditetapkan pengguna kustom, berdasarkan apakah tes yang ditentukan berlalu.

Setiap item dalam`Features`terdiri dari parameter-parameter berikut:

`Name`

Nama fitur.

`Value`

Tidak wajib. Nilai kustom yang ingin Anda gunakan dalam laporan, dan bukan`supported`. Jika nilai ini tidak ditentukan, IDT kemudian berdasarkan menetapkan nilai fitur untuk`supported`atau`not-supported`berdasarkan hasil pengujian. Jika Anda menguji fitur yang sama dengan kondisi yang berbeda, Anda dapat menggunakan nilai kustom untuk setiap instance fitur tersebut di`Features`daftar, dan IDT menggabungkan nilai fitur untuk kondisi yang didukung. Untuk informasi selengkapnya, lihat

`Condition`

Ekspresi konteks yang mengevaluasi ke`Boolean`nilai. Jika nilai yang dievaluasi benar, IDT menambahkan fitur ke laporan pengujian setelah selesai menjalankan test suite. Jika nilai yang dievaluasi adalah palsu, tes tidak termasuk dalam laporan.

`Tests`

Tidak wajib. Daftar deskriptor tes. Semua tes yang ditentukan dalam daftar ini harus lulus pada fitur yang akan didukung.

Setiap deskriptor pengujian dalam daftar ini menggunakan ID grup uji dan satu atau beberapa ID kasus uji untuk mengidentifikasi pengujian individual yang dijalankan dari kelompok uji tertentu. Deskriptor pengujian menggunakan format berikut:

```
GroupId: group-id
CaseIds: [test-id, test-id] # optional
```

Anda harus menentukan`Tests`atau`OneOfTests`untuk setiap fitur di`Features`daftar.

`OneOfTests`

Tidak wajib. Daftar deskriptor tes. Setidaknya satu tes yang ditentukan dalam daftar ini harus lulus pada fitur yang akan didukung.

Setiap deskriptor pengujian dalam daftar ini menggunakan ID grup uji dan satu atau beberapa ID kasus uji untuk mengidentifikasi pengujian individual yang dijalankan dari kelompok uji tertentu. Deskriptor pengujian menggunakan format berikut:

```
GroupId: group-id
CaseIds: [test-id, test-id] # optional
```

Anda harus menentukan`Tests`atau`OneOfTests`untuk setiap fitur di`Features`daftar.

`IsRequired`

Nilai boolean yang mendefinisikan apakah fitur diperlukan dalam laporan pengujian. Nilai default-nya adalah `false`.

Example

Konteks orkestrasi uji

Konteks pengujian adalah dokumen JSON baca-saja yang berisi data yang tersedia untuk orkestrasi uji selama eksekusi. Konteks orkestrasi uji hanya dapat diakses dari orkestrasi uji, 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 orkestrasi pengujian menggunakan format berikut:

```
{
    "pool": {
        <device-json-pool-element>
    },
    "userData": {
        <userdata-json-content>
    },
    "config": {
        <config-json-content>
    }
}
```

`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 di file `config.json`.

Anda dapat melakukan kueri atas konteks tersebut dengan menggunakan notasi JSONPath. Sintaks untuk kueri JSONPath dalam definisi keadaan adalah `{{`*`query`*`}}`. Ketika Anda mengakses data dari konteks pengujian, pastikan bahwa setiap nilai dievaluasi pada string, angka, atauBoolean.

Untuk informasi lebih lanjut tentang peggunaan notasi JSONPath untuk mengakses data dari konteks, lihat Gunakan konteks IDT.

Konfigurasikan state machine IDT

> **Important**
>
> Mulai di IDT v4.5.1, mesin negara ini tidak digunakan lagi. Kami sangat menyarankan Anda untuk menggunakan orkestrator uji baru. Untuk informasi selengkapnya, lihat Konfigurasikan orkestrasi uji IDT.

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.

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 `state_machine.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 `Defaultke 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`test.json` file, 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 eksekusi tes yang diperlukan berdasarkan perintah dan argumen yang ditentukan dalam`test.json` file, 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
- IDT Client SDK for Java

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

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, orkestrator tes untuk rangkaian tes Anda harus menyertakan serangkaian`Choice` keadaan`RunTask` dan keadaan tertentu dalam orkestrator pengujian 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 orkestrator kustom, gunakan[Contoh mesin status: Jalankan grup uji yang dipilih pengguna](#) sebagai contoh untuk menambahkan keadaan yang diperlukan dalam orkestrator tes 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 dan log IDT

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:

Batas waktu

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.

Interupsi

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 `test_orchestrator.yaml`, Anda menggunakan `{{query}}`.

  Jika Anda menggunakan mesin negara usang, maka di`state_machine.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 yang terletak di *<device-tester-extract-location>*`/configs/` folder. 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`

Opsional. 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 disediakan dari konteks IDT dengan `devicePool.features` kueri, atau dari konteks orkestrator pengujian 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`

Opsional. 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`

Opsional. 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 `ssh`.

`connectivity.containerUser`

Opsional. 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 `ssh`.

> **Note**
>
> Untuk memeriksa apakah pelari pengujian mengonfigurasi koneksi perangkat yang salah untuk pengujian, Anda dapat mengambil

`pool.Devices[0].Connectivity.Protocol` dari konteks orkestrator pengujian dan membandingkannya dengan nilai yang diharapkan dalam status. `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-version>",
                "jobSlots": <job-slots>
```

```
            }
        ],
        "devices": [
            {
                "id": "<device-id>",
                "connectivity": {
                    "protocol": "ssh | uart | docker",
                    // ssh
                    "ip": "<ip-address>",
                    "port": <port-number>,
                    "auth": {
                        "method": "pki | password",
                        "credentials": {
                            "user": "<user-name>",
                            // pki
                            "privKeyPath": "/path/to/private/key",

                            // password
                            "password": "<password>",
                        }
                    },

                    // uart
                    "serialPort": "<serial-port>",

                    // docker
                    "containerId": "<container-id>",
                    "containerUser": "<container-user-name>",
                }
            }
        ]
    }
]
```

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`

Opsional. 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`

Opsional. 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`

Opsional. 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 `ssh`.

`connectivity.containerUser`

Opsional. 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 `ssh`.

(Opsional) Konfigurasikan config.json

File `config.json` berisi informasi konfigurasi untuk IDT. Biasanya, test runner tidak perlu mengubah file ini kecuali untuk memberikan kredensial pengguna AWS untuk IDT, dan secara opsional, sebuah wilayah AWS. Jika kredensial AWS dengan izin yang diperlukan disediakan, AWS IoT Device Tester akan 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 jalur dalam file ini didefinisikan relatif terhadap *< device-tester-extract-location >*.

`log.location`

Jalur 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`

Opsional. 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
```

Saat Anda menjalankan IDT dalam mode debug, IDT sebenarnya tidak meluncurkan rangkaian pengujian atau menjalankan orkestrator pengujian; sebaliknya, IDT berinteraksi dengan IDE Anda untuk merespons permintaan yang dibuat dari rangkaian pengujian 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 orkestrator pengujian dan tidak akan menghasilkan file laporan apa pun. 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. Lihat informasi yang lebih lengkap di Jalankan IDT dalam mode debug.

Tinjau hasil tes dan log IDT

Bagian ini menjelaskan format di mana IDT menghasilkan log konsol dan laporan uji.

Format pesan konsol

AWS IoT Device Tester menggunakan format standar untuk mencetak pesan ke konsol ketika memulai rangkaian tes. 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`Reportstatus`di orkestrator tes 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
```

Skema laporan AWS IoT Device Tester

`awsiotdevicetester_report.xml` adalah laporan bertanda tangan 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 memberikan AWS kredensil dengan izin yang diperlukan, AWS IoT Device Tester kumpulkan dan kirimkan metrik penggunaan ke. AWS Ini adalah fitur opt-in dan digunakan untuk meningkatkan fungsi IDT. IDT mengumpulkan informasi seperti berikut:

- Akun AWS ID yang digunakan untuk menjalankan IDT
- AWS CLI Perintah IDT yang digunakan untuk menjalankan tes
- Suite pengujian 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. Cukup jangan menyimpan AWS kredensil Anda, dan jika Anda memiliki AWS kredensil yang disimpan, jangan konfigurasikan `config.json` file untuk mengaksesnya.

Konfigurasikan AWS kredensil 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 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”.

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

Untuk mendaftar untuk Akun AWS

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

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

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

Untuk membuat pengguna administrator, pilih salah satu opsi berikut.

Pilih salah satu cara untuk mengelola administrator Anda	Untuk	Oleh	Anda juga bisa
Di Pusat Identitas IAM (Direkomendasikan)	Gunakan kredensi jangka pendek untuk mengakses. AWS Ini sejalan dengan praktik terbaik keamanan. Untuk informasi tentang praktik terbaik, lihat Praktik terbaik keamanan di IAM di	Mengikuti petunjuk di Memulai di Panduan AWS IAM Identity Center Pengguna.	Konfigurasikan akses terprogram dengan Mengonfigurasi AWS CLI yang akan digunakan AWS IAM Identity Center dalam AWS Command Line Interface Panduan Pengguna.

Pilih salah satu cara untuk mengelola administrator Anda	Untuk	Oleh	Anda juga bisa
	Panduan Pengguna IAM.		
Di IAM (Tidak direkomendasikan)	Gunakan kredensi jangka panjang untuk mengakses. AWS	Mengikuti petunjuk dalam Membuat pengguna admin IAM pertama Anda dan grup pengguna di Panduan Pengguna IAM.	Konfigurasikan akses terprogram dengan Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

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 or AWS Command Line Interface (AWS CLI) untuk membuat kebijakan IAM dan pengguna untuk IDT, lalu melampirkan kebijakan ke 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 Tinjau kebijakan.

d. Untuk Nama, masukkan **`IDTUsageMetricsIAMPermissions`**. Di bawah Ringkasan, tinjau izin yang diberikan oleh kebijakan Anda.

e. 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 Atur izin, pilih Lampirkan kebijakan yang ada ke pengguna 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 AWS kredensi Anda.

Untuk mengonfigurasi izin untuk IDT (AWS CLI)

Ikuti langkah-langkah ini untuk menggunakan AWS CLI untuk mengkonfigurasi izin untuk IDT untuk. AWS IoT Greengrass

1. Di komputer Anda, instal dan konfigurasikan AWS CLI jika belum diinstal. Ikuti langkah-langkah di Menginstal AWS CLI di Panduan Pengguna AWS Command Line Interface .

   > **Note**
   >
   > AWS CLI Ini adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan dari shell baris perintah Anda.

2. Buat kebijakan terkelola pelanggan berikut yang memberikan izin untuk mengelola IDT dan peran. AWS IoT Greengrass

   Linux or Unix

   ```
   aws iam create-policy --policy-name IDTUsageMetricsIAMPermissions --policy-
   document '{
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "iot-device-tester:SendMetrics"
               ],
               "Resource": "*"
           }
       ]
   }'
   ```

   Windows command prompt

   ```
   aws iam create-policy --policy-name IDTUsageMetricsIAMPermissions --policy-
   document
   ```

```
                                                '{\"Version\": \"2012-10-17\",
 \"Statement\": [{\"Effect\": \"Allow\", \"Action\": [\"iot-device-
tester:SendMetrics\"], \"Resource\": \"*\"}]}'
```

> **Note**
>
> Langkah ini mencakup contoh prompt perintah Windows karena menggunakan sintaks JSON yang berbeda dari perintah terminal Linux, MacOS, atau Unix.

PowerShell

```
aws iam create-policy --policy-name IDTUsageMetricsIAMPermissions --policy-
document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot-device-tester:SendMetrics"
            ],
            "Resource": "*"
        }
    ]
}'
```

3. 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 AWS kredensi Anda.

Memberikan AWS kredensi ke IDT

Untuk mengizinkan IDT mengakses AWS kredensil Anda dan mengirimkan metrik AWS, lakukan hal berikut:

1. Simpan AWS kredensil untuk pengguna IAM Anda sebagai variabel lingkungan atau dalam file kredensil:

   a. Untuk menggunakan variabel lingkungan, jalankan perintah berikut.

      Linux or Unix

      ```
      export AWS_ACCESS_KEY_ID=access-key
      export AWS_SECRET_ACCESS_KEY=secret-access-key
      ```

      Windows Command Prompt (CMD)

      ```
      set AWS_ACCESS_KEY_ID=access-key
      set AWS_SECRET_ACCESS_KEY=secret-access-key
      ```

      PowerShell

      ```
      $env:AWS_ACCESS_KEY_ID="access-key"
      $env:AWS_SECRET_ACCESS_KEY="secret-access-key"
      ```

   b. Untuk menggunakan file kredensial, tambahkan informasi berikut ke file. `~/.aws/credentials`

      ```
      [profile-name]
      aws_access_key_id=access-key
      aws_secret_access_key=secret-access-key
      ```

2. Konfigurasikan bagian `auth` dari file `config.json`. Untuk informasi selengkapnya, lihat (Opsional) Konfigurasikan config.json.

Penyelesaian masalah IDT untuk V2 AWS IoT Greengrass

IDT untuk V2 AWS IoT Greengrass menuliskan kesalahan ke berbagai lokasi berdasarkan jenis kesalahan. IDT menuliskan kesalahan ke konsol, file log, dan laporan tes.

Di mana mencari kesalahan

Kesalahan tingkat tinggi ditampilkan di konsol saat pengujian sedang berjalan, dan ringkasan pengujian yang gagal ditampilkan saat semua pengujian selesai.`awsiotdevicetester_report.xml`berisi ringkasan semua kesalahan yang menyebabkan tes gagal. IDT menyimpan file log untuk setiap pengujian yang dijalankan di direktori dengan UUID untuk eksekusi pengujian, ditampilkan di konsol selama uji coba.

Direktori log uji IDT adalah *<device-tester-extract-location>*`/results/`*<execution-id>*`/logs/`. Direktori ini berisi file-file berikut yang ditampilkan dalam tabel. Ini berguna untuk debugging.

File	Deskripsi
`test_manager.log`	Log ditulis ke konsol saat tes sedang berjalan. Ringkasan hasil pada akhir file ini mencakup daftar tes yang gagal. Catatan peringatan dan kesalahan dalam file ini dapat memberikan beberapa informasi tentang kegagalan.
test-group-id `/`*test-case-id* `/`*test-name* `.log`	Log terperrinci untuk tes spesifik dalam grup uji. Untuk tes yang men-deploy komponen Greengrass, uji kasus file log tersebut disebut `greengrass-test-run.log` .
test-group-id `/`*test-case-id* `/greengrass.log`	Log terperinci untuk perangkat lunak inti AWS IoT Greengrass. IDT menyalin file ini dari perangkat yang diuji ketika menjalankan tes yang menginstal perangkat lunak inti AWS IoT Greengrass pada perangkat. Untuk informasi selengkapnya tentang pesan dalam file log ini,

File	Deskripsi
	lihat Pemecahan masalah AWS IoT Greengrass V2.
test-group-id /*test-case-id*/*component-name* .log	Log terperinci untuk komponen Greengrass yang di-deploy selama uji coba. IDT menyalin file log komponen dari perangkat yang diuji ketika menjalankan tes yang men-deploy komponen tertentu. Nama setiap file log komponen sesuai dengan nama komponen yang di-deploy. Untuk informasi selengkapnya tentang pesan dalam berkas log ini, lihat Pemecahan masalah AWS IoT Greengrass V2.

Menyelesaikan IDT untuk kesalahan V2 AWS IoT Greengrass

Sebelum Anda menjalankan IDT untuk AWS IoT Greengrass, dapatkan file konfigurasi yang benar di tempatnya. Jika Anda menerima parsing dan konfigurasi kesalahan, langkah pertama Anda adalah menemukan dan menggunakan templat konfigurasi yang sesuai untuk lingkungan Anda.

Jika Anda masih mengalami masalah, lihat proses debugging berikut.

Topik

- Kesalahan resolusi alias
- Kesalahan konflik
- Tidak dapat memulai kesalahan uji
- Gambar kualifikasi docker ada kesalahan
- Gagal membaca kredensi
- Kesalahan Guice dengan PreInstalled Greengrass
- Pengecualian tanda tangan tidak valid
- Kesalahan kualifikasi machine learning
- Penerapan gagal Open Test Framework (OTF)
- Kesalahan parsing

- Kesalahan ditolak izin
- Kesalahan pembuatan laporan kualifikasi
- Parameter yang diperlukan kehilangan kesalahan
- Pengecualian keamanan di macOS
- Kesalahan koneksi SSH
- Kesalahan kualifikasi stream manager
- Kesalahan batas waktu
- Kesalahan pemeriksaan versi

Kesalahan resolusi alias

Saat Anda menjalankan suite pengujian khusus, Anda mungkin melihat kesalahan berikut di konsol dan di`test_manager.log`.

```
Couldn't resolve placeholders: couldn't do a json lookup: index out of range
```

Kesalahan ini dapat terjadi ketika alias yang dikonfigurasi dalam orkestrator pengujian IDT tidak diselesaikan dengan benar atau jika nilai yang diselesaikan tidak ada dalam file konfigurasi. Untuk mengatasi kesalahan ini, pastikan bahwa`device.json`dan`userdata.json` berisi informasi yang benar yang diperlukan untuk rangkaian pengujian Anda. Untuk informasi tentang konfigurasi yang diperlukan untukAWS IoT Greengrasskualifikasi, lihatKonfigurasikan pengaturan IDT untuk menjalankan rangkaian AWS IoT Greengrass kualifikasi.

Kesalahan konflik

Anda mungkin melihat kesalahan berikut ketika Anda menjalankan rangkaian kualifikasi AWS IoT Greengrass secara bersamaan di lebih dari satu perangkat.

```
ConflictException: Component [com.example.IDTHelloWorld : 1.0.0] for account [account-
id] already exists with state: [DEPLOYABLE] { RespMetadata: { StatusCode: 409,
 RequestID: “id” }, Message_: “Component [com.example.IDTHelloWorld : 1.0.0] for
 account [account-id] already exists with state: [DEPLOYABLE]” }
```

Eksekusi uji bersamaan belum didukung untuk rangkaian kualifikasi AWS IoT Greengrass. Jalankan rangkaian kualifikasi secara berurutan untuk setiap perangkat.

Tidak dapat memulai kesalahan uji

Anda mungkin mengalami kesalahan yang mengarah ke kegagalan yang terjadi ketika tes mencoba untuk dimulai. Ada beberapa kemungkinan penyebabnya, jadi lakukan hal berikut:

- Pastikan bahwa nama kolam dalam perintah eksekusi Anda benar-benar ada. IDT mengacu nama kolam secara langsung dari file `device.json`.
- Pastikan bahwa perangkat di kolam Anda memiliki parameter konfigurasi yang benar.

Gambar kualifikasi docker ada kesalahan

Tes kualifikasi manajer aplikasi Docker menggunakan`amazon/amazon-ec2-metadata-mock`gambar kontainer di Amazon ECR untuk memenuhi syarat perangkat yang diuji.

Anda mungkin menerima kesalahan berikut jika gambar sudah ada dalam kontainer Docker pada perangkat yang diuji.

```
The Docker image amazon/amazon-ec2-metadata-mock:version already exists on the device.
```

Jika Anda sebelumnya telah mengunduh gambar ini dan menjalankan `amazon/amazon-ec2-metadata-mock` di perangkat Anda, pastikan Anda menghapus gambar ini dari perangkat yang sedang diuji sebelum Anda menjalankan tes kualifikasi.

Gagal membaca kredensi

Saat menguji perangkat Windows, Anda mungkin menemukan`Failed to read credential`kesalahan dalam`greengrass.log`berkas jika pengguna yang Anda gunakan untuk menyambung ke perangkat yang sedang diuji tidak diatur di pengelola kredensi pada perangkat tersebut.

Untuk mengatasi kesalahan ini, konfigurasikan pengguna dan kata sandi untuk pengguna IDT di pengelola kredensi pada perangkat yang sedang diuji.

Untuk informasi selengkapnya, lihat Konfigurasikan kredensional pengguna untuk perangkat Windows.

Kesalahan Guice dengan PreInstalled Greengrass

Saat menjalankan IDT dengan PreInstalled Greengrass, jika Anda menemukan kesalahan`Guice`atau`ErrorInCustomProvider`, periksa apakah

file`userdata.json`memiliki`InstalledDirRootOnDevice`atur ke folder instalasi Greengrass. IDT memeriksa file`effectiveConfig.yaml`di bawah`<InstallationDirRootOnDevice>/config/effectiveConfig.yaml`.

Untuk informasi selengkapnya, lihat Konfigurasikan kredensional pengguna untuk perangkat Windows.

Pengecualian tanda tangan tidak valid

Ketika Anda menjalankan tes kualifikasi Lambda, Anda mungkin menemukan`invalidsignatureexception`kesalahan jika mesin host IDT Anda mengalami masalah akses jaringan. Setel ulang router Anda dan jalankan tes lagi.

Kesalahan kualifikasi machine learning

Saat menjalankan tes kualifikasi machine learning (ML), Anda mungkin mengalami kegagalan kualifikasi jika perangkat Anda tidak memenuhipersyaratanuntuk menyebarkanAWS-menyediakan komponen ML. Untuk memecahkan masalah kesalahan kualifikasi ML, lakukan hal berikut:

- Cari rincian kesalahan dalam log komponen untuk komponen yang di-deploy selama uji coba. Log komponen terletak di direktori *<device-tester-extract-location>*/results/*<execution-id>*/logs/*<test-group-id>*.
- Tambahkan argumen `-Dgg.persist=installed.software` ke file `test.json` untuk uji kasus yang gagal. File `test.json` terletak di *<device-tester-extract-location>*/tests/GGV2Q_*version* directory.

Penerapan gagal Open Test Framework (OTF)

Jika pengujian OTF gagal menyelesaikan penerapan, kemungkinan penyebabnya adalah izin yang ditetapkan untuk folder indukTempResourcesDirOnDevicedanInstallationDirRootOnDevice. Untuk mengatur izin folder ini dengan benar, jalankan perintah berikut. Ganti*folder-name*dengan nama folder induk.

```
sudo chmod 755 folder-name
```

Kesalahan parsing

Kesalahan ketik dalam konfigurasi JSON dapat menyebabkan kesalahan parsing. Sering kali, masalah ini adalah akibat dari menghilangkan kurung, koma, atau tanda kutip dari file JSON Anda.

IDT melakukan validasi JSON dan mencetak informasi debugging. IDT mencetak garis di mana kesalahan terjadi, nomor baris, dan nomor kolom kesalahan sintaks. Informasi ini akan cukup untuk membantu Anda memperbaiki kesalahan, tetapi jika Anda masih tidak dapat menemukan kesalahan, Anda dapat melakukan validasi secara manual di IDE Anda, editor teks seperti Atom atau Sublime, atau melalui alat online seperti JSONLint.

Kesalahan ditolak izin

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 menyarankan Anda menggunakan sudo visudo saat Anda mengedit `/etc/sudoers`.

Kesalahan pembuatan laporan kualifikasi

IDT mendukung empat versi *major*.*minor* terakhir dari versi rangkaian kualifikasi V2 AWS IoT Greengrass (GGV2Q) untuk menghasilkan laporan kualifikasi yang dapat Anda kirimkan ke AWS Partner Network untuk menyertakan perangkat Anda di Katalog Perangkat AWS Partner. Versi sebelumnya dari rangkaian kualifikasi ini tidak menghasilkan laporan kualifikasi.

Jika Anda memiliki pertanyaan tentang kebijakan dukungan, hubungi AWS Support.

Parameter yang diperlukan kehilangan kesalahan

Ketika IDT menambahkan fitur baru, ia mungkin memperkenalkan perubahan pada file konfigurasi. Penggunaan file konfigurasi lama mungkin akan merusak konfigurasi Anda. Jika hal ini terjadi, file `<test_case_id>.log` di bawah `/results/<execution-id>/logs` secara eksplisit mencantumkan semua parameter yang hilang. IDT juga memvalidasi skema file konfigurasi JSON Anda untuk memverifikasi bahwa Anda menggunakan versi terbaru yang didukung.

Pengecualian keamanan di macOS

Saat Anda menjalankan IDT di komputer host macOS, IDT memblokir IDT agar tidak berjalan. Untuk menjalankan IDT, berikan pengecualian keamanan ke executable yang merupakan bagian dari fungsionalitas runtime IDT. Saat Anda melihat tampilan pesan peringatan di komputer host Anda, lakukan hal berikut untuk setiap executable yang berlaku:

Untuk memberikan pengecualian keamanan untuk executable IDT

1. Di komputer MacOS, pada menu Apple, bukaPreferensi Sistem.
2. Pilih Keamanan & Privasi, kemudian pada tab Umum, pilih ikon kunci untuk membuat perubahan pada pengaturan keamanan.
3. Dalam kasus diblokir`devicetester_mac_x86-64`, cari pesanny`a"devicetester_mac_x86-64" was blocked from use because it is not from an identified developer.`dan pilihIzinkan Pokoknya.
4. Lanjutkan pengujian IDT, sampai Anda melewati semua executable yang terlibat.

Kesalahan koneksi SSH

Ketika IDT tidak dapat terhubung ke perangkat yang diuji, ia mencatat kegagalan koneksi di `/results/<execution-id>/logs/<test-case-id>.log`. Pesan SSH muncul di bagian atas file log ini karena penyambungan ke perangkat yang diuji adalah salah satu operasi pertama yang dilakukan oleh IDT.

Sebagian besar konfigurasi Windows menggunakan aplikasi terminal PuTTy untuk terhubung ke host Linux. Aplikasi ini mengharuskan Anda mengonversi file kunci privat PEM standar ke dalam format Windows berpemilik yang disebut PPK. Jika Anda mengonfigurasi SSH di `device.json`, gunakan

file PEM. Jika Anda menggunakan file PPK, IDT tidak dapat membuat koneksi SSH dengan AWS IoT Greengrass Anda dan tidak dapat menjalankan tes.

Dimulai dengan IDT v4.4.0, jika Anda belum mengaktifkan SFTP di perangkat yang sedang diuji, Anda mungkin melihat kesalahan berikut di file log.

```
SSH connection failed with EOF
```

Untuk mengatasi kesalahan ini, aktifkan SFTP di perangkat Anda.

Kesalahan kualifikasi stream manager

Ketika Anda menjalankan uji kualifikasi manajer pengaliran, Anda mungkin akan melihat kesalahan berikut di file `com.aws.StreamManagerExport.log`.

```
Failed to upload data to S3
```

Kesalahan ini dapat terjadi ketika stream manager menggunakan kredensial AWS di file `~/root/.aws/credentials` pada perangkat Anda dan tidak menggunakan kredensial lingkungan yang diekspor oleh IDT ke perangkat yang sedang diuji. Untuk mencegah masalah ini, hapus file `credentials` di perangkat Anda, dan jalankan kembali tes kualifikasi.

Kesalahan batas waktu

Anda dapat meningkatkan batas waktu untuk setiap tes dengan menentukan pengganda batas waktu yang diterapkan ke nilai default dari setiap batas waktu 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 GGV2Q_1.0.0 --pool-id DevicePool1 --timeout-
multiplier 2.5
```

Untuk informasi lebih lanjut, jalankan `run-suite --help`.

Beberapa kesalahan batas waktu terjadi ketika kasus uji IDT tidak dapat diselesaikan karena masalah konfigurasi. Anda tidak dapat mengatasi kesalahan ini dengan meningkatkan pengganda batas waktu. Gunakan log dari uji coba untuk memecahkan masalah konfigurasi yang mendasarinya.

- Jika log komponen MQTT atau Lambda berisi`Access denied`kesalahan, folder instalasi Greengrass Anda mungkin tidak memiliki izin file yang benar. Jalankan perintah berikut untuk setiap folder di jalur instalasi yang Anda tentukan di`userdata.json`berkas.

  ```
  sudo chmod 755 folder-name
  ```

- Jika log Greengrass menunjukkan bahwa penerapan Greengrass CLI belum selesai, lakukan hal berikut:
  - Verifikasi bahwa`bash`diinstal pada perangkat yang sedang diuji.
  - Jika Anda`userdata.json`file termasuk`GreengrassCliVersion`parameter konfigurasi, hapus. Parameter ini tidak digunakan lagi di IDT v4.1.0 dan versi yang lebih baru. Untuk informasi selengkapnya, lihat Konfigurasikan userdata.json.
- Jika pengujian penerapan Lambda gagal dengan pesan kesalahan "Memvalidasi Lambda publish: time out" dan Anda menerima kesalahan dalam file log pengujian (`idt-gg2-lambda-function-idt-<resource-id>.log`) yang mengatakan`Error: Could not find or load main class com.amazonaws.greengrass.runtime.LambdaRuntime.`, lakukan hal berikut:
  - Verifikasi folder apa yang digunakan`InstallationDirRootOnDevice`di`userdata.json`berkas.
  - Pastikan izin pengguna yang benar diatur di perangkat Anda. Untuk lebih jelasnya, lihatKonfigurasikan izin pengguna di perangkat Anda.

Kesalahan pemeriksaan versi

IDT mengeluarkan kesalahan berikut ketika kredensial pengguna AWS untuk pengguna IDT tidak memiliki izin IAM yang diperlukan.

```
Failed to check version compatibility
```

Pengguna AWS yang tidak memiliki izin IAM yang diperlukan.

Kebijakan Support AWS IoT Device Tester untuk AWS IoT Greengrass

AWS IoT Device Tester for AWS IoT Greengrass adalah alat otomatisasi pengujian yang digunakan untuk memvalidasi dan memenuhi syarat AWS IoT Greengrass perangkat Anda untuk dimasukkan

dalam Katalog AWS Partner Perangkat. Kami menyarankan Anda menggunakan versi terbaru AWS IoT Greengrass dan AWS IoT Device Tester untuk menguji atau memenuhi syarat perangkat Anda.

Setidaknya satu versi AWS IoT Device Tester tersedia untuk setiap versi yang didukung AWS IoT Greengrass. Untuk versi yang didukung AWS IoT Greengrass, lihat Versi inti Greengrass. Untuk versi yang didukung AWS IoT Device Tester, lihatVersi yang didukung AWS IoT Device Tester untuk AWS IoT Greengrass V2.

Anda juga dapat menggunakan salah satu versi yang didukung dari AWS IoT Greengrass dan AWS IoT Device Tester untuk menguji atau memenuhi syarat perangkat Anda. Meskipun Anda dapat terus menggunakan versi yang tidak didukung AWS IoT Device Tester, versi tersebut tidak menerima perbaikan bug atau pembaruan. Jika Anda memiliki pertanyaan tentang kebijakan dukungan, hubungi AWS Support.

Solusi IoT berbasis Greengrass

Everyware Eurotech GreenEdge sedang dalam rilis pratinjau untuk AWS IoT Greengrass dan dapat berubah sewaktu-waktu. Solusi ini tidak didukung olehAWS. Anda harus menghubungi Eurotech untuk masalah apa pun dengan perangkat ini.

AWS IoT Greengrassmenawarkan solusi dari Mitra untuk mengoptimalkan pengalaman Anda menginstal Greengrass. Berikut ini adalah solusi yang AWS telah bermitra dengan Eurotech untuk ditawarkan. Solusi ini dilengkapi dengan runtime AWS IoT Greengrass Core edge dan kemampuan tambahan yang sudah diinstal sebelumnya.

Eurotech

AWStelah bermitra dengan Eurotech untuk menawarkan solusi IoT bagi pelanggan yang mencari perangkat yang dilengkapi dengan AWS IoT Greengrass perangkat lunak Core yang sudah diinstal sebelumnya. Eurotech's Everyware GreenEdge adalah perangkat lunak IoT edge yang telah dikonfigurasi sebelumnya dan pra-kualifikasi oleh. AWS Solusi ini menggabungkan kemampuan Greengrass dan Eurotech Everyware Software Framework (ESF) untuk menawarkan pelanggan konektivitas selatan yang luas melalui adaptor protokol seperti: Modbus, OPC-UA Client/Server, S7, TwinCat, J1939, DNP3 Master/Outstation, dan banyak lagi. Dengan solusi ini, Anda juga dapat mengirim data ke AWS Cloud dan terhubung ke semua AWS layanan ke utara (sepertiAWS IoT Core,,, Amazon S3 AWS IoT SiteWiseAWS IoT Analytics, dan Amazon Kinesis Video Streams). Dikombinasikan dengan Everyware Cloud, solusi manajemen perangkat Eurotech, solusi ini memperkenalkan layanan Penyediaan Zero-Touch baru, yang menyederhanakan orientasi perangkat dan penyebaran massal.

Untuk informasi lebih lanjut tentang Eurotech, lihat Eurotech.

Pemecahan masalah AWS IoT Greengrass V2

Gunakan informasi dan solusi pemecahan masalah di bagian ini untuk membantu menyelesaikan masalah. AWS IoT Greengrass Version 2

Topik

Lihat perangkat lunak AWS IoT Greengrass inti dan log komponen

Perangkat lunak AWS IoT Greengrass Core menulis log ke sistem file lokal yang dapat Anda gunakan untuk melihat informasi real-time tentang perangkat inti. Anda juga dapat mengonfigurasi perangkat inti untuk menulis CloudWatch log ke Log, sehingga Anda dapat memecahkan masalah perangkat inti dari jarak jauh. Log ini dapat membantu Anda mengidentifikasi masalah dengan komponen, penerapan, dan perangkat inti. Untuk informasi selengkapnya, lihat Memantau AWS IoT Greengrass log.

AWS IoT Greengrass Masalah perangkat lunak inti

Memecahkan masalah perangkat lunak AWS IoT Greengrass inti.

Topik

- Tidak dapat memulai perangkat lunak AWS IoT Greengrass Inti sebagai layanan sistem
- Tidak dapat mengatur inti sebagai layanan sistem
- Tidak dapat terhubung ke AWS IoT Core
- Kesalahan kehabisan memori
- Tidak dapat menginstal Greengrass CLI
- User root is not allowed to execute
- com.aws.greengrass.lifecyclemanager.GenericExternalService: Could not determine user/group to run with
- Failed to map segment from shared object: operation not permitted
- Gagal mengatur layanan Windows
- com.aws.greengrass.util.exceptions.TLSAuthException: Failed to get trust manager
- com.aws.greengrass.deployment.IotJobsHelper: No connection available during subscribing to Iot Jobs descriptions topic. Will retry in sometime
- software.amazon.awssdk.services.iam.model.IamException: The security token included in the request is invalid
- software.amazon.awssdk.services.iot.model.IotException: User: <user> is not authorized to perform: iot:GetPolicy
- Error: com.aws.greengrass.shadowmanager.sync.model.FullShadowSyncRequest: Could not execute cloud shadow get request
- Operation aws.greengrass#<operation> is not supported by Greengrass
- java.io.FileNotFoundException: <stream-manager-store-root-dir>/stream_manager_metadata_store (Permission denied)
- com.aws.greengrass.security.provider.pkcs11.PKCS11CryptoKeyService: Private key or certificate with label <label> does not exist
- software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException: User: <user> is not authorized to perform: secretsmanager:GetSecretValue on resource: <arn>
- software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException: Access to KMS is not allowed
- java.lang.NoClassDefFoundError: com/aws/greengrass/security/CryptoKeySpi
- com.aws.greengrass.security.provider.pkcs11.PKCS11CryptoKeyService: CKR_OPERATION_NOT_INITIALIZED
- Greengrass core device stuck on nucleus v2.12.3

Tidak dapat mengatur perangkat inti

Jika penginstal perangkat lunak AWS IoT Greengrass Core gagal dan Anda tidak dapat menyiapkan perangkat inti, Anda mungkin perlu menghapus instalasi perangkat lunak dan mencoba lagi. Untuk informasi selengkapnya, lihat Hapus instalasi perangkat lunak inti AWS IoT Greengrass.

Tidak dapat memulai perangkat lunak AWS IoT Greengrass Inti sebagai layanan sistem

Jika perangkat lunak AWS IoT Greengrass Core gagal memulai, periksa log layanan sistem untuk mengidentifikasi masalah. Salah satu masalah umum adalah di mana Java tidak tersedia pada variabel lingkungan PATH (Linux) atau variabel sistem PATH (Windows).

Tidak dapat mengatur inti sebagai layanan sistem

Anda mungkin melihat kesalahan ini ketika penginstal perangkat lunak AWS IoT Greengrass Core gagal diatur AWS IoT Greengrass sebagai layanan sistem. Pada perangkat Linux, kesalahan ini biasanya terjadi jika perangkat inti tidak memiliki sistem init systemd. Installer dapat berhasil mengatur perangkat lunak AWS IoT Greengrass Core bahkan jika gagal untuk mengatur layanan sistem.

Lakukan salah satu hal berikut:

- Konfigurasikan dan jalankan perangkat lunak AWS IoT Greengrass Core sebagai layanan sistem. Anda harus mengkonfigurasi perangkat lunak sebagai layanan sistem untuk menggunakan semua fitur AWS IoT Greengrass. Anda dapat menginstal systemd atau menggunakan sistem init yang berbeda. Untuk informasi selengkapnya, lihat Konfigurasikan inti Greengrass sebagai layanan sistem.
- Jalankan perangkat lunak AWS IoT Greengrass inti tanpa layanan sistem. Anda dapat menjalankan perangkat lunak menggunakan skrip loader yang disiapkan penginstal di folder root Greengrass. Untuk informasi selengkapnya, lihat Jalankan perangkat lunak inti AWS IoT Greengrass tanpa layanan sistem.

Tidak dapat terhubung ke AWS IoT Core

Anda mungkin melihat kesalahan ini ketika perangkat lunak AWS IoT Greengrass Core tidak dapat terhubung AWS IoT Core untuk mengambil pekerjaan penerapan, misalnya. Lakukan hal-hal berikut:

- Periksa apakah perangkat inti Anda dapat terhubung ke internet dan AWS IoT Core. Untuk informasi selengkapnya tentang AWS IoT Core titik akhir yang terhubung dengan perangkat Anda, lihatKonfigurasikan perangkat lunak AWS IoT Greengrass Inti.
- Periksa apakah perangkat inti Anda AWS IoT menggunakan sertifikat yang memungkinkan`iot:Connect`,, `iot:Publish``iot:Receive`, dan `iot:Subscribe` izin.
- Jika perangkat inti Anda menggunakan proksi jaringan, periksa apakah perangkat inti Anda memiliki peran perangkat dan apakah perannya memungkinkan izin `iot:Connect`, `iot:Publish`, `iot:Receive`, dan `iot:Subscribe`.

Kesalahan kehabisan memori

Kesalahan ini biasanya terjadi jika perangkat Anda tidak memiliki memori yang cukup untuk mengalokasikan objek di tumpukan Java. Pada perangkat dengan memori terbatas, Anda mungkin perlu menentukan ukuran tumpukan maksimum untuk mengontrol alokasi memori. Untuk informasi selengkapnya, lihat Kontrol alokasi memori dengan opsi JVM.

Tidak dapat menginstal Greengrass CLI

Anda mungkin melihat pesan konsol berikut saat menggunakan `--deploy-dev-tools` argumen dalam perintah instalasi untuk AWS IoT Greengrass Core.

```
Thing group exists, it could have existing deployment and devices, hence NOT creating
 deployment for Greengrass first party dev tools, please manually create a deployment
 if you wish to
```

Hal ini terjadi ketika komponen Greengrass CLI tidak diinstal karena perangkat inti Anda adalah anggota dari grup objek yang memiliki deployment yang ada. Jika Anda melihat pesan ini, Anda dapat secara manual menyebarkan komponen Greengrass CLI (`aws.greengrass.Cli`) ke perangkat untuk menginstal Greengrass CLI. Untuk informasi selengkapnya, lihat Instal Greengrass CLI.

User root is not allowed to execute

Anda mungkin melihat kesalahan ini ketika pengguna yang menjalankan perangkat lunak AWS IoT Greengrass Core (biasanya`root`) tidak memiliki izin untuk menjalankan `sudo` dengan pengguna dan grup apa pun. Untuk pengguna sistem `ggc_user` default, kesalahan ini terlihat seperti berikut:

```
Sorry, user root is not allowed to execute <command> as ggc_user:ggc_group.
```

Periksa bahwa file `/etc/sudoers` Anda memberikan izin pengguna untuk menjalankan `sudo` sebagai kelompok lain. Izin untuk pengguna di `/etc/sudoers` seharusnya terlihat seperti contoh berikut.

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

com.aws.greengrass.lifecyclemanager.GenericExternalService: Could not determine user/group to run with

Anda mungkin melihat kesalahan ini ketika perangkat inti mencoba menjalankan komponen, dan inti Greengrass tidak menentukan pengguna sistem default yang akan digunakan untuk menjalankan komponen.

Untuk memperbaiki masalah ini, konfigurasikan inti Greengrass untuk menentukan pengguna sistem default yang menjalankan komponen. Lihat informasi yang lebih lengkap di Konfigurasikan pengguna yang menjalankan komponen dan Konfigurasikan pengguna komponen default.

Failed to map segment from shared object: operation not permitted

Anda mungkin melihat kesalahan ini ketika perangkat lunak AWS IoT Greengrass Inti gagal memulai karena `/tmp` folder dipasang dengan `noexec` izin. Pustaka AWS Common Runtime (CRT) menggunakan `/tmp` folder secara default.

Lakukan salah satu hal berikut:

- Jalankan perintah berikut untuk memasang kembali `/tmp` folder dengan `exec` izin dan coba lagi.

  ```
  sudo mount -o remount,exec /tmp
  ```

- Jika Anda menjalankan Greengrass nucleus v2.5.0 atau yang lebih baru, Anda dapat menyetel opsi JVM untuk mengubah folder yang digunakan pustaka CRT. AWS Anda dapat menentukan `jvmOptions` parameter dalam konfigurasi komponen inti Greengrass dalam penerapan atau saat Anda menginstal perangkat lunak Core. AWS IoT Greengrass Ganti */path/to/use* dengan path ke folder yang dapat digunakan perpustakaan CRT. AWS

  ```
  {
  ```

```
  "jvmOptions": "-Daws.crt.lib.dir=\"/path/to/use\""
}
```

Gagal mengatur layanan Windows

Anda mungkin melihat kesalahan ini jika Anda menginstal perangkat lunak AWS IoT Greengrass Core pada perangkat Microsoft Windows 2016. Perangkat lunak AWS IoT Greengrass Core tidak didukung pada Windows 2016, untuk daftar sistem operasi yang didukung, lihatPlatform yang didukung.

Jika Anda harus menggunakan Windows 2016, Anda dapat melakukan hal berikut:

1. Buka zip arsip instalasi AWS IoT Greengrass Core yang diunduh
2. Di `Greengrass` direktori buka `bin/greengrass.xml.template` file.
3. Tambahkan `<autoRefresh>` tag ke akhir file tepat sebelum `</service>` tag.

```
  </log>
  <autoRefresh>false</autoRefresh>
</service>
```

com.aws.greengrass.util.exceptions.TLSAuthException: Failed to get trust manager

Anda mungkin melihat kesalahan ini saat menginstal perangkat lunak AWS IoT Greengrass Core tanpa file root certificate authority (CA).

```
2022-06-05T10:00:39.556Z [INFO] (main) com.aws.greengrass.lifecyclemanager.Kernel:
 service-loaded. {serviceName=DeploymentService}
2022-06-05T10:00:39.943Z [WARN] (main)
 com.aws.greengrass.componentmanager.ClientConfigurationUtils: configure-greengrass-
mutual-auth. Error during configure greengrass client mutual auth. {}
com.aws.greengrass.util.exceptions.TLSAuthException: Failed to get trust manager
```

Periksa apakah Anda menentukan file CA root yang valid dengan `rootCaPath` parameter dalam file konfigurasi yang Anda berikan kepada penginstal. Untuk informasi selengkapnya, lihat Instal perangkat lunak inti AWS IoT Greengrass.

com.aws.greengrass.deployment.IotJobsHelper: No connection available during subscribing to Iot Jobs descriptions topic. Will retry in sometime

Anda mungkin melihat pesan peringatan ini ketika perangkat inti tidak dapat terhubung AWS IoT Core untuk berlangganan pemberitahuan pekerjaan penerapan. Lakukan hal-hal berikut:

- Periksa apakah perangkat inti terhubung ke internet dan dapat mencapai titik akhir AWS IoT data yang Anda konfigurasikan. Untuk informasi selengkapnya tentang titik akhir yang digunakan perangkat inti, lihatIzinkan lalu lintas perangkat melalui proxy atau firewall.
- Periksa log Greengrass untuk kesalahan lain yang mengungkapkan akar penyebab lainnya.

software.amazon.awssdk.services.iam.model.IamException: The security token included in the request is invalid

Anda mungkin melihat kesalahan ini saat menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan otomatis, dan penginstal menggunakan token AWS sesi yang tidak valid. Lakukan hal-hal berikut:

- Jika Anda menggunakan kredensil keamanan sementara, periksa apakah token sesi sudah benar dan Anda menyalin dan menempelkan token sesi lengkap.
- Jika Anda menggunakan kredensil keamanan jangka panjang, periksa apakah perangkat tidak memiliki token sesi dari waktu di mana Anda sebelumnya menggunakan kredensil sementara. Lakukan hal-hal berikut:
  1. Jalankan perintah berikut untuk menghapus variabel lingkungan token sesi.

     Linux or Unix

     ```
     unset AWS_SESSION_TOKEN
     ```

     Windows Command Prompt (CMD)

     ```
     set AWS_SESSION_TOKEN=
     ```

     PowerShell

     ```
     Remove-Item Env:\AWS_SESSION_TOKEN
     ```

2. Periksa apakah file AWS kredensialnya,`~/.aws/credentials`, berisi token sesi,. `aws_session_token` Jika demikian, hapus baris itu dari file.

```
aws_session_token = AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
```

Anda juga dapat menginstal perangkat lunak AWS IoT Greengrass Core tanpa memberikan AWS kredensional. Untuk informasi selengkapnya, lihat Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan sumber daya manual atau Instal perangkat lunak AWS IoT Greengrass Core dengan penyediaan AWS IoT armada.

software.amazon.awssdk.services.iot.model.IotException: User: <user> is not authorized to perform: iot:GetPolicy

Anda mungkin melihat kesalahan ini saat menginstal perangkat lunak AWS IoT Greengrass Core dengan penyediaan otomatis, dan penginstal menggunakan AWS kredensil yang tidak memiliki izin yang diperlukan. Untuk informasi selengkapnya tentang izin yang diperlukan, lihatKebijakan IAM minimal untuk penginstal untuk menyediakan sumber daya.

Periksa izin untuk identitas IAM kredensil, dan berikan identitas IAM izin yang diperlukan yang hilang.

Error: com.aws.greengrass.shadowmanager.sync.model.FullShadowSyncRequest: Could not execute cloud shadow get request

Anda mungkin melihat kesalahan ini saat menggunakan komponen pengelola bayangan untuk menyinkronkan bayangan perangkat dengan AWS IoT Core. Kode status HTTP 403 menunjukkan bahwa kesalahan ini terjadi karena AWS IoT kebijakan perangkat inti tidak memberikan izin untuk memanggil`GetThingShadow`.

```
com.aws.greengrass.shadowmanager.sync.model.FullShadowSyncRequest: Could not execute
 cloud shadow get request. {thing name=MyGreengrassCore, shadow name=MyShadow}
2021-07-14T21:09:02.456Z [ERROR] (pool-2-thread-109)
 com.aws.greengrass.shadowmanager.sync.SyncHandler: sync. Skipping sync request. {thing
 name=MyGreengrassCore, shadow name=MyShadow}
com.aws.greengrass.shadowmanager.exception.SkipSyncRequestException:
 software.amazon.awssdk.services.iotdataplane.model.IotDataPlaneException:
```

```
null (Service: IotDataPlane, Status Code: 403, Request ID:
f6e713ba-1b01-414c-7b78-5beb3f3ad8f6, Extended Request ID: null)
```

Untuk menyinkronkan bayangan lokal dengan AWS IoT Core, AWS IoT kebijakan perangkat inti harus memberikan izin berikut:

- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

Periksa AWS IoT kebijakan perangkat inti, dan tambahkan izin yang diperlukan yang tidak ada. Untuk informasi selengkapnya, lihat hal berikut:

- AWS IoT Core tindakan kebijakan dalam Panduan AWS IoT Pengembang
- Memperbarui AWS IoT kebijakan perangkat inti

Operation aws.greengrass#<operation> is not supported by Greengrass

Anda mungkin melihat kesalahan ini saat menggunakan operasi komunikasi antarproses (IPC) dalam komponen Greengrass kustom, dan komponen yang AWS disediakan yang diperlukan tidak diinstal pada perangkat inti.

Untuk memperbaiki masalah ini, tambahkan komponen yang diperlukan sebagai dependensi dalam resep komponen Anda, sehingga perangkat lunak AWS IoT Greengrass Core menginstal komponen yang diperlukan saat Anda menerapkan komponen Anda.

- Mengambil nilai rahasia — `aws.greengrass.SecretManager`
- Berinteraksi dengan bayangan lokal — `aws.greengrass.ShadowManager`
- Mengelola penerapan dan komponen lokal — `aws.greengrass.Cli` v2.6.0 atau yang lebih baru
- Mengautentikasi dan mengotorisasi perangkat klien - `aws.greengrass.clientdevices.Auth` v2.2.0 atau yang lebih baru

java.io.FileNotFoundException: <stream-manager-store-root-dir>/stream_manager_metadata_store (Permission denied)

Anda mungkin melihat kesalahan ini di file log pengelola aliran (`aws.greengrass.StreamManager.log`) saat mengonfigurasi pengelola aliran untuk menggunakan folder root yang tidak ada atau memiliki izin yang benar. Untuk informasi selengkapnya tentang cara mengonfigurasi folder ini, lihat konfigurasi manajer aliran.

com.aws.greengrass.security.provider.pkcs11.PKCS11CryptoKeyService: Private key or certificate with label <label> does not exist

Kesalahan ini terjadi ketika komponen penyedia PKCS #11 tidak dapat menemukan atau memuat kunci pribadi atau sertifikat yang Anda tentukan saat Anda mengonfigurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan modul keamanan perangkat keras (HSM). Lakukan hal-hal berikut:

- Periksa apakah kunci pribadi dan sertifikat disimpan di HSM menggunakan slot, PIN pengguna, dan label objek yang Anda konfigurasikan perangkat lunak AWS IoT Greengrass Core untuk digunakan.
- Periksa apakah kunci pribadi dan sertifikat menggunakan label objek yang sama di HSM.
- Jika HSM Anda mendukung ID objek, periksa apakah kunci pribadi dan sertifikat menggunakan ID objek yang sama di HSM.

Periksa dokumentasi untuk HSM Anda untuk mempelajari cara menanyakan detail tentang token keamanan di HSM. Jika Anda perlu mengubah slot, label objek, atau ID objek untuk token keamanan, periksa dokumentasi untuk HSM Anda untuk mempelajari cara melakukannya.

software.amazon.awssdk.services.secretsmanager.model.SecretsManagerExcep User: <user> is not authorized to perform: secretsmanager:GetSecretValue on resource: <arn>

Kesalahan ini dapat terjadi ketika Anda menggunakan komponen manajer rahasia untuk menyebarkan AWS Secrets Manager rahasia. Jika IAM role pertukaran token perangkat inti tidak memberikan izin untuk mendapatkan rahasia, deployment itu gagal dan log Greengrass mencakup kesalahan ini.

Untuk mengotorisasi perangkat inti untuk mengunduh rahasia

1. Tambahkan izin `secretsmanager:GetSecretValue` ke peran pertukaran token perangkat inti. Contoh pernyataan kebijakan berikut memberikan izin untuk mendapatkan nilai rahasia.

   ```
   {
       "Effect": "Allow",
       "Action": [
           "secretsmanager:GetSecretValue"
       ],
       "Resource": [
           "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyGreengrassSecret-
   abcdef"
       ]
   }
   ```

   Untuk informasi selengkapnya, lihat Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

2. Terapkan kembali deployment ke perangkat inti. Lakukan salah satu hal berikut:

   - Revisi deployment tanpa perubahan apa pun. Perangkat inti mencoba untuk mengunduh rahasia lagi ketika menerima deployment yang direvisi. Untuk informasi selengkapnya, lihat Revisi deployment.
   - Mulai ulang perangkat lunak AWS IoT Greengrass Core untuk mencoba lagi penyebaran. Lihat informasi yang lebih lengkap di Jalankan perangkat lunak inti AWS IoT Greengrass

   Deployment berhasil jika secret manager berhasil mengunduh rahasia tersebut.

software.amazon.awssdk.services.secretsmanager.model.SecretsManagerExcep Access to KMS is not allowed

Kesalahan ini dapat terjadi ketika Anda menggunakan komponen manajer rahasia untuk menyebarkan AWS Secrets Manager rahasia yang dienkripsi oleh kunci. AWS Key Management Service Jika peran IAM pertukaran token perangkat inti tidak memberikan izin untuk mendekripsi rahasia, penerapan gagal dan log Greengrass menyertakan kesalahan ini.

Untuk memperbaiki masalah ini, tambahkan `kms:Decrypt` izin ke peran pertukaran token perangkat inti. Untuk informasi selengkapnya, lihat hal berikut:

- Enkripsi rahasia dan dekripsi dalam Panduan Pengguna AWS Secrets Manager
- Otorisasi perangkat inti untuk berinteraksi dengan AWS layanan

java.lang.NoClassDefFoundError: com/aws/greengrass/security/CryptoKeySpi

Anda mungkin melihat kesalahan ini ketika mencoba menginstal perangkat lunak AWS IoT Greengrass Core dengan keamanan perangkat keras dan Anda menggunakan versi inti Greengrass sebelumnya yang tidak mendukung integrasi keamanan perangkat keras. Untuk menggunakan integrasi keamanan perangkat keras, Anda harus menggunakan Greengrass nucleus v2.5.3 atau yang lebih baru.

com.aws.greengrass.security.provider.pkcs11.PKCS11CryptoKeyService: CKR_OPERATION_NOT_INITIALIZED

Anda mungkin melihat kesalahan ini saat menggunakan pustaka TPM2 saat menjalankan AWS IoT Greengrass Core sebagai layanan sistem.

Kesalahan ini menunjukkan bahwa Anda perlu menambahkan variabel lingkungan yang menyediakan lokasi penyimpanan PKCS #11 di file layanan systemd AWS IoT Greengrass Core.

Untuk informasi selengkapnya, lihat bagian Persyaratan pada dokumentasi Penyedia PKCS #11 komponen.

Greengrass core device stuck on nucleus v2.12.3

Jika perangkat inti Greengrass Anda tidak akan merevisi penerapan Anda dari nucleus versi 2.12.3, Anda mungkin perlu mengunduh dan mengganti file dengan Greengrass nucleus versi 2.12.2. `Greengrass.jar` Lakukan hal-hal berikut:

1. Pada perangkat inti Greengrass Anda, jalankan perintah berikut untuk menghentikan perangkat lunak Greengrass Core.

   Linux or Unix

   ```
   sudo systemctl stop greengrass
   ```

Windows Command Prompt (CMD)

```
sc stop "greengrass"
```

PowerShell

```
Stop-Service -Name "greengrass"
```

2. Di perangkat inti Anda, unduh AWS IoT Greengrass perangkat lunak ke file bernama`greengrass-2.12.2.zip`.

   Linux or Unix

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-2.12.2.zip >
    greengrass-2.12.2.zip
   ```

   Windows Command Prompt (CMD)

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-2.12.2.zip >
    greengrass-2.12.2.zip
   ```

   PowerShell

   ```
   iwr -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-2.12.2.zip -
   OutFile greengrass-2.12.2.zip
   ```

3. Buka zip perangkat lunak AWS IoT Greengrass Core ke folder di perangkat Anda. Ganti *`GreengrassInstaller`*dengan folder yang ingin Anda gunakan.

   Linux or Unix

   ```
   unzip greengrass-2.12.2.zip -d GreengrassInstaller && rm greengrass-2.12.2.zip
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir GreengrassInstaller && tar -xf greengrass-2.12.2.zip -
   C GreengrassInstaller && del greengrass-2.12.2.zip
   ```

PowerShell

```
Expand-Archive -Path greengrass-2.12.2.zip -DestinationPath .\
\GreengrassInstaller
rm greengrass-2.12.2.zip
```

4. Jalankan perintah berikut untuk mengganti file JAR Greengrass versi 2.12.3 nukleus dengan file JAR Greengrass versi nukleus 2.12.2.

   Linux or Unix

   ```
   sudo cp ./GreengrassInstaller/lib/Greengrass.jar /greengrass/v2/packages/
   artifacts-unarchived/aws.greengrass.Nucleus/2.12.3/aws.greengrass.nucleus/lib
   ```

   Windows Command Prompt (CMD)

   ```
   robocopy ./GreengrassInstaller/lib/Greengrass.jar /greengrass/v2/packages/
   artifacts-unarchived/aws.greengrass.Nucleus/2.12.3/aws.greengrass.nucleus/lib /E
   ```

   PowerShell

   ```
   cp -Path ./GreengrassInstaller/lib/Greengrass.jar -Destination /
   greengrass/v2/packages/artifacts-unarchived/aws.greengrass.Nucleus/2.12.3/
   aws.greengrass.nucleus/lib
   ```

5. Jalankan perintah berikut untuk memulai perangkat lunak Greengrass Core.

   Linux or Unix

   ```
   sudo systemctl start greengrass
   ```

   Windows Command Prompt (CMD)

   ```
   sc start "greengrass"
   ```

   PowerShell

   ```
   Start-Service -Name "greengrass"
   ```

AWS IoT Greengrass masalah cloud

Gunakan informasi berikut untuk memecahkan masalah dengan AWS IoT Greengrass konsol dan API. Setiap entri sesuai dengan pesan kesalahan yang mungkin Anda lihat ketika Anda melakukan tindakan.

An error occurred (AccessDeniedException) when calling the CreateComponentVersion operation: User: arn:aws:iam::123456789012:user/<username> is not authorized to perform: null

Anda mungkin melihat kesalahan ini saat membuat versi komponen dari AWS IoT Greengrass konsol atau dengan CreateComponentVersionoperasi.

Kesalahan ini menunjukkan bahwa resep Anda bukan JSON atau YAML yang valid. Periksa sintaks resep Anda, perbaiki masalah sintaks, dan coba lagi. Anda dapat menggunakan pemeriksa sintaks JSON atau YAML online untuk mengidentifikasi masalah sintaks dalam resep Anda.

Invalid Input: Encountered following errors in Artifacts: {<s3ArtifactUri> = Specified artifact resource cannot be accessed}

Anda mungkin melihat kesalahan ini saat membuat versi komponen dari AWS IoT Greengrass konsol atau dengan CreateComponentVersionoperasi. Kesalahan ini menunjukkan bahwa artefak S3 dalam resep komponen tidak valid.

Lakukan hal-hal berikut:

- Periksa apakah bucket S3 berada di tempat yang sama Wilayah AWS di mana Anda membuat komponen. AWS IoT Greengrass tidak mendukung permintaan lintas wilayah untuk artefak komponen.
- Periksa apakah URI artefak adalah URL objek S3 yang valid, dan periksa artefak yang ada di URL objek S3 tersebut.
- Periksa apakah Anda Akun AWS memiliki izin untuk mengakses artefak di URL objek S3-nya.

INACTIVE deployment status

Anda mungkin mendapatkan status `INACTIVE` penerapan saat memanggil ListDeploymentsAPI tanpa AWS IoT kebijakan dependen yang diperlukan. Anda harus memiliki izin yang diperlukan untuk mendapatkan status penerapan yang akurat. Anda dapat menemukan tindakan dependen dengan melihat di Tindakan yang ditentukan oleh AWS IoT Greengrass V2 dan mengikuti izin yang diperlukan`ListDeployments`. Tanpa AWS IoT izin dependen yang diperlukan, Anda masih akan melihat status penerapan tetapi Anda mungkin melihat status penerapan yang tidak akurat. `INACTIVE`

Masalah deployment perangkat inti

Memecahkan masalah penerapan pada perangkat inti Greengrass. Setiap entri sesuai dengan pesan log yang mungkin Anda lihat di perangkat inti Anda.

Topik

- Error: com.aws.greengrass.componentmanager.exceptions.PackageDownloadException: Failed to download artifact
- Error: com.aws.greengrass.componentmanager.exceptions.ArtifactChecksumMismatchException: Integrity check for downloaded artifact failed. Probably due to file corruption.
- Error: com.aws.greengrass.componentmanager.exceptions.NoAvailableComponentVersionException: Failed to negotiate component <name> version with cloud and no local applicable version satisfying requirement <requirements>
- software.amazon.awssdk.services.greengrassv2data.model.ResourceNotFoundException: The latest version of Component <componentName> doesn't claim platform <coreDevicePlatform> compatibility
- com.aws.greengrass.componentmanager.exceptions.PackagingException: The deployment attempts to update the nucleus from aws.greengrass.Nucleus-<version> to aws.greengrass.Nucleus-<version> but no component of type nucleus was included as target component
- Error: com.aws.greengrass.deployment.exceptions.DeploymentException: Unable to process deployment. Greengrass launch directory is not set up or Greengrass is not set up as a system service

- Info: com.aws.greengrass.deployment.exceptions.RetryableDeploymentDocumentDownloadException: Greengrass Cloud Service returned an error when getting full deployment configuration
- Warn: com.aws.greengrass.deployment.DeploymentService: Failed to get thing group hierarchy
- Info: com.aws.greengrass.deployment.DeploymentDocumentDownloader: Calling Greengrass cloud to get full deployment configuration
- Caused by: software.amazon.awssdk.services.greengrassv2data.model.GreengrassV2DataException: null (Service: GreengrassV2Data, Status Code: 403, Request ID: <some_request_id>, Extended Request ID: null)

Error: com.aws.greengrass.componentmanager.exceptions.PackageDownloadExceptio Failed to download artifact

Anda mungkin melihat kesalahan ini ketika perangkat lunak AWS IoT Greengrass Core gagal mengunduh artefak komponen saat perangkat inti menerapkan penerapan. Deployment gagal sebagai akibat dari kesalahan ini.

Ketika Anda menerima kesalahan ini, log juga mencakup jejak tumpukan yang dapat Anda gunakan untuk mengidentifikasi masalah tertentu. Masing-masing entri berikut sesuai dengan pesan yang mungkin Anda lihat di jejak tumpukan pesan kesalahan `Failed to download artifact`.

Topik

- software.amazon.awssdk.services.s3.model.S3Exception: null (Service: S3, Status Code: 403, Request ID: null, ...)
- software.amazon.awssdk.services.s3.model.S3Exception: Access Denied (Service: S3, Status Code: 403, Request ID: <requestID>

software.amazon.awssdk.services.s3.model.S3Exception: null (Service: S3, Status Code: 403, Request ID: null, ...)

PackageDownloadException Kesalahan mungkin termasuk jejak tumpukan ini dalam kasus berikut:

- Artefak komponen tidak tersedia di URL objek S3 yang Anda tentukan dalam resep komponen. Pastikan Anda mengunggah artefak ke bucket S3 dan URI artefak cocok dengan URL objek S3 dari artefak di bucket.
- Peran pertukaran token perangkat inti tidak mengizinkan perangkat lunak AWS IoT Greengrass Core mengunduh artefak komponen dari URL objek S3 yang Anda tentukan dalam resep komponen. Periksa apakah peran pertukaran token memungkinkan `s3:GetObject` URL objek S3 tempat artefak tersedia.

software.amazon.awssdk.services.s3.model.S3Exception: Access Denied (Service: S3, Status Code: 403, Request ID: <requestID>

PackageDownloadException Kesalahan mungkin termasuk jejak tumpukan ini ketika perangkat inti tidak memiliki izin untuk menelepon`s3:GetBucketLocation`. Pesan kesalahan juga menyertakan pesan berikut.

```
reason: Failed to determine S3 bucket location
```

Periksa apakah peran pertukaran token perangkat inti memungkinkan `s3:GetBucketLocation` bucket S3 tempat artefak tersedia.

Error: com.aws.greengrass.componentmanager.exceptions.ArtifactChecksumMismatchI Integrity check for downloaded artifact failed. Probably due to file corruption.

Anda mungkin melihat kesalahan ini ketika perangkat lunak AWS IoT Greengrass Core gagal mengunduh artefak komponen saat perangkat inti menerapkan penerapan. Penerapan gagal karena checksum file artefak yang diunduh tidak cocok dengan checksum yang AWS IoT Greengrass dihitung saat Anda membuat komponen.

Lakukan hal-hal berikut:

- Periksa apakah file artefak berubah dalam bucket S3 tempat Anda meng-hosting-nya. Jika file berubah sejak Anda membuat komponen, pulihkan ke versi sebelumnya yang diharapkan perangkat inti. Jika Anda tidak dapat memulihkan file ke versi sebelumnya, atau jika Anda ingin menggunakan versi baru dari file itu, buat versi baru dari komponen dengan file artefak.
- Periksa koneksi internet perangkat inti Anda. Kesalahan ini dapat terjadi jika file artefak rusak saat mengunduh. Buat deployment baru dan coba lagi.

Error: com.aws.greengrass.componentmanager.exceptions.NoAvailableComponentVers Failed to negotiate component <name> version with cloud and no local applicable version satisfying requirement <requirements>

Anda mungkin melihat kesalahan ini bila perangkat inti tidak dapat menemukan versi komponen yang memenuhi persyaratan penyebaran untuk perangkat inti tersebut. Perangkat inti memeriksa komponen dalam AWS IoT Greengrass layanan dan pada perangkat lokal. Pesan kesalahan mencakup target setiap penyebaran dan persyaratan versi penyebaran untuk komponen. Target deployment dapat berupa objek, grup objek, atau `LOCAL_DEPLOYMENT`, yang mewakili deployment lokal pada perangkat inti.

Masalah ini dapat terjadi pada kasus berikut:

- Perangkat inti adalah target dari beberapa penyebaran yang memiliki persyaratan versi komponen yang bertentangan. Sebagai contoh, perangkat inti tersebut mungkin menjadi target dari beberapa deployment yang mencakup komponen `com.example.HelloWorld`, di mana satu deployment memerlukan versi 1.0.0 dan yang lain memerlukan versi 1.0.1. Tidak mungkin memiliki komponen yang memenuhi kedua persyaratan tersebut, sehingga deployment itu gagal.
- Versi komponen tidak ada di AWS IoT Greengrass layanan atau di perangkat lokal. Komponen mungkin telah dihapus, misalnya.
- Ada versi komponen yang memenuhi persyaratan versi, tetapi tidak ada yang kompatibel dengan platform perangkat inti.
- AWS IoT Kebijakan perangkat inti tidak memberikan `greengrass:ResolveComponentCandidates` izin. Cari `Status Code: 403` di log kesalahan untuk mengidentifikasi masalah ini. Untuk mengatasi masalah ini, tambahkan izin `greengrass:ResolveComponentCandidates` ke kebijakan perangkat inti AWS IoT . Untuk informasi selengkapnya, lihat Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2.

Untuk mengatasi masalah ini, revisi penyebaran dengan menyertakan versi komponen yang kompatibel atau hapus yang tidak kompatibel. Untuk informasi selengkapnya tentang cara merevisi penyebaran cloud, lihat Revisi deployment. Untuk informasi selengkapnya tentang cara merevisi deployment lokal, lihat perintah AWS IoT Greengrass Buat CLI deployment.

software.amazon.awssdk.services.greengrassv2data.model.ResourceNotFoundE The latest version of Component <componentName> doesn't claim platform <coreDevicePlatform> compatibility

Anda mungkin melihat kesalahan ini saat menerapkan komponen ke perangkat inti, dan komponen tersebut tidak mencantumkan platform yang kompatibel dengan platform perangkat inti. Lakukan salah satu hal berikut:

- Jika komponennya adalah komponen Greengrass kustom, Anda dapat memperbarui komponen agar kompatibel dengan perangkat inti. Tambahkan manifes baru yang cocok dengan platform perangkat inti, atau perbarui manifes yang ada agar sesuai dengan platform perangkat inti. Untuk informasi selengkapnya, lihat AWS IoT Greengrass referensi resep komponen.
- Jika komponen disediakan oleh AWS, periksa apakah versi lain dari komponen tersebut kompatibel dengan perangkat inti. Jika tidak ada versi yang kompatibel, hubungi kami untuk AWS re:Postmenggunakan AWS IoT Greengrass tag, atau kontak AWS Support.

com.aws.greengrass.componentmanager.exceptions.PackagingException: The deployment attempts to update the nucleus from aws.greengrass.Nucleus-<version> to aws.greengrass.Nucleus-<version> but no component of type nucleus was included as target component

Anda mungkin melihat kesalahan ini saat menerapkan komponen yang bergantung pada inti Greengrass, dan perangkat inti menjalankan versi inti Greengrass sebelumnya daripada versi minor terbaru yang tersedia. Kesalahan ini terjadi karena perangkat lunak AWS IoT Greengrass Core mencoba memperbarui komponen secara otomatis ke versi terbaru yang kompatibel. Namun, perangkat lunak AWS IoT Greengrass Core mencegah inti Greengrass memperbarui ke versi minor baru, karena AWS beberapa komponen yang disediakan bergantung pada versi minor tertentu dari inti Greengrass. Untuk informasi selengkapnya, lihat Perilaku pembaruan inti Greengrass.

Anda harus merevisi penerapan untuk menentukan versi inti Greengrass yang ingin Anda gunakan. Lakukan salah satu hal berikut:

- Merevisi penerapan untuk menentukan versi inti Greengrass yang saat ini dijalankan perangkat inti.
- Merevisi penerapan untuk menentukan versi minor selanjutnya dari inti Greengrass. Jika Anda memilih opsi ini, Anda juga harus memperbarui versi semua komponen yang AWS disediakan yang

bergantung pada versi minor tertentu dari inti Greengrass. Untuk informasi selengkapnya, lihat Komponen yang disediakan oleh AWS.

Error: com.aws.greengrass.deployment.exceptions.DeploymentException: Unable to process deployment. Greengrass launch directory is not set up or Greengrass is not set up as a system service

Anda mungkin melihat kesalahan ini saat memindahkan perangkat Greengrass dari satu grup hal ke grup lainnya, lalu kembali ke grup asli dengan penerapan yang memerlukan Greengrass untuk memulai ulang.

Untuk mengatasi masalah ini, buat ulang direktori peluncuran untuk perangkat. Kami juga sangat menyarankan untuk meningkatkan ke versi 2.9.6 atau yang lebih baru dari inti Greengrass.

Berikut ini adalah skrip Linux untuk membuat ulang direktori peluncuran. Simpan skrip dalam file bernama`fix_directory.sh`.

```
#!/bin/bash

set -e

GG_ROOT=$1
GG_VERSION=$2

CURRENT="$GG_ROOT/alts/current"

if [ ! -L "$CURRENT" ]; then
  mkdir -p $GG_ROOT/alts/directory_fix
  echo "Relinking $GG_ROOT/alts/directory_fix to $CURRENT"
  ln -sf $GG_ROOT/alts/directory_fix $CURRENT
fi

TARGET=$(readlink $CURRENT)

if [[ ! -d "$TARGET" ]]; then
  echo "Creating directory: $TARGET"
  mkdir -p "$TARGET"
fi

DISTRO_LINK="$TARGET/distro"
```

```
DISTRO="$GG_ROOT/packages/artifacts-unarchived/aws.greengrass.Nucleus/$GG_VERSION/
aws.greengrass.nucleus/"
echo "Relinking Nucleus artifacts to $DISTRO_LINK"
ln -sf $DISTRO $DISTRO_LINK
```

Untuk menjalankan skrip, jalankan perintah berikut:

```
[root@ip-172-31-27-165 ~]# ./fix_directory.sh /greengrass/v2 2.9.5
Relinking /greengrass/v2/alts/directory_fix to /greengrass/v2/alts/current
Relinking Nucleus artifacts to /greengrass/v2/alts/directory_fix/distro
```

Info: com.aws.greengrass.deployment.exceptions.RetryableDeploymentDocumentDow Greengrass Cloud Service returned an error when getting full deployment configuration

Anda mungkin melihat kesalahan ini ketika perangkat inti menerima dokumen deployment yang besar, yang merupakan dokumen deployment yang lebih besar dari 7 KB (untuk deployment yang menargetkan objek) atau 31 KB (untuk deployment yang menargetkan grup objek). Untuk mengambil dokumen penerapan besar, AWS IoT kebijakan perangkat inti harus mengizinkan izin tersebut. `greengrass:GetDeploymentConfiguration` Kesalahan ini dapat terjadi ketika perangkat inti tidak memiliki izin ini. Ketika kesalahan ini terjadi, deployment tersebut mencoba ulang tanpa batas waktu, dan statusnya Dalam proses (`IN_PROGRESS`).

Untuk mengatasi masalah ini, tambahkan `greengrass:GetDeploymentConfiguration` izin ke AWS IoT kebijakan perangkat inti. Untuk informasi selengkapnya, lihat Memperbarui AWS IoT kebijakan perangkat inti.

Warn: com.aws.greengrass.deployment.DeploymentService: Failed to get thing group hierarchy

Anda mungkin melihat peringatan ini saat perangkat inti menerima penerapan dan AWS IoT kebijakan perangkat inti tidak mengizinkan `greengrass:ListThingGroupsForCoreDevice` izin tersebut. Saat Anda membuat penerapan, perangkat inti menggunakan izin ini untuk mengidentifikasi grup benda dan menghapus komponen untuk grup apa pun dari mana Anda menghapus perangkat inti. Jika perangkat inti menjalankan Greengrass nucleus v2.5.0, penerapan gagal. Jika perangkat inti menjalankan Greengrass nucleus v2.5.1 atau yang lebih baru, penerapan berlangsung tetapi tidak

menghapus komponen. Untuk informasi selengkapnya tentang perilaku penghapusan grup benda, lihatDeploy komponen AWS IoT Greengrass ke perangkat.

Untuk memperbarui perilaku perangkat inti untuk menghapus komponen untuk grup benda tempat Anda menghapus perangkat inti, tambahkan `greengrass:ListThingGroupsForCoreDevice` izin ke AWS IoT kebijakan perangkat inti. Untuk informasi selengkapnya, lihat Memperbarui AWS IoT kebijakan perangkat inti.

Info: com.aws.greengrass.deployment.DeploymentDocumentDownloader: Calling Greengrass cloud to get full deployment configuration

Anda mungkin melihat pesan informasi ini dicetak beberapa kali tanpa kesalahan, karena perangkat inti mencatat kesalahan di tingkat log `DEBUG`. Masalah ini dapat terjadi ketika perangkat inti menerima dokumen deployment yang besar. Ketika kesalahan ini terjadi, deployment ini mencoba ulang tanpa batas waktu, dan statusnya Dalam proses (`IN_PROGRESS`). Untuk informasi selengkapnya tentang cara mengatasi masalah ini, lihat entri pemecahan masalah ini.

Caused by: software.amazon.awssdk.services.greengrassv2data.model.GreengrassV2DataE null (Service: GreengrassV2Data, Status Code: 403, Request ID: <some_request_id>, Extended Request ID: null)

Anda mungkin melihat kesalahan ini ketika API dataplane tidak memiliki `iot:Connect` izin. Jika Anda tidak memiliki kebijakan yang benar, Anda akan menerima`GreengrassV2DataException: 403`. Untuk membuat kebijakan izin, ikuti petunjuk berikut:Buat AWS IoT kebijakan.

Masalah komponen perangkat inti

Memecahkan masalah komponen Greengrass pada perangkat inti.

Topik

- Warn: '<command>' is not recognized as an internal or external command
- Skrip Python tidak mencatat pesan
- Konfigurasi komponen tidak diperbarui saat mengubah konfigurasi default
- awsiot.greengrasscoreipc.model.UnauthorizedError

- com.aws.greengrass.authorization.exceptions.AuthorizationException: Duplicate policy ID "<id>" for principal "<componentList>"
- com.aws.greengrass.tes.CredentialRequestHandler: Error in retrieving AwsCredentials from TES (HTTP 400)
- com.aws.greengrass.tes.CredentialRequestHandler: Error in retrieving AwsCredentials from TES (HTTP 403)
- com.aws.greengrass.tes.CredentialsProviderError: Could not load credentials from any providers
- Received error when attempting to retrieve ECS metadata: Could not connect to the endpoint URL: "<tokenExchangeServiceEndpoint>"
- copyFrom: <configurationPath> is already a container, not a leaf
- com.aws.greengrass.componentmanager.plugins.docker.exceptions.DockerLoginException: Error logging into the registry using credentials - 'The stub received bad data.'
- java.io.IOException: Cannot run program "cmd" ...: [LogonUser] The password for this account has expired.
- aws.greengrass.StreamManager: Instant exceeds minimum or maximum instant

Warn: '<command>' is not recognized as an internal or external command

Anda mungkin melihat kesalahan ini di log komponen Greengrass ketika AWS IoT Greengrass perangkat lunak Core gagal menjalankan perintah dalam skrip siklus hidup komponen. Status komponen menjadi `BROKEN` sebagai akibat dari kesalahan ini. Kesalahan ini dapat terjadi jika pengguna sistem yang menjalankan komponen, sepertiggc_user, tidak dapat menemukan perintah yang dapat dieksekusi di folder di PATH.

Pada perangkat Windows, periksa apakah folder yang berisi executable ada di `PATH` untuk pengguna sistem yang menjalankan komponen. Jika hilang dari`PATH`, lakukan salah satu hal berikut:

- Tambahkan folder executable ke variabel `PATH` sistem, yang tersedia untuk semua pengguna. Kemudian, restart komponen.

  Jika Anda menjalankan Greengrass nucleus 2.5.0, setelah Anda memperbarui `PATH` variabel sistem, Anda harus memulai ulang perangkat lunak Core untuk menjalankan komponen AWS IoT Greengrass dengan yang diperbarui. `PATH` Jika perangkat lunak AWS IoT Greengrass Core tidak menggunakan yang diperbarui `PATH` setelah Anda memulai ulang perangkat lunak, restart perangkat dan coba lagi. Untuk informasi selengkapnya, lihat Jalankan perangkat lunak inti AWS IoT Greengrass.

- Tambahkan folder executable ke variabel PATH pengguna untuk pengguna sistem yang menjalankan komponen.

Skrip Python tidak mencatat pesan

Perangkat inti Greengrass mengumpulkan log yang dapat Anda gunakan untuk mengidentifikasi masalah dengan komponen. Jika pesan `stdout` dan `stderr` skrip Python Anda tidak muncul di log komponen Anda, Anda mungkin perlu menyiram buffer atau menonaktifkan buffer untuk stream output standar ini di Python. Lakukan salah satu langkah berikut ini:

- Jalankan Python dengan -u untuk menonaktifkan buffering pada `stdout` dan `stderr`.

  Linux or Unix

  ```
  python3 -u hello_world.py
  ```

  Windows

  ```
  py -3 -u hello_world.py
  ```

- Gunakan Setenv dalam resep komponen Anda untuk mengatur variabel lingkungan PYTHONUNBUFFERED pada string tak kosong. Variabel lingkungan ini menonaktifkan buffering pada `stdout` dan `stderr`.
- Siram buffer untuk pengaliran `stdout` atau `stderr`. Lakukan salah satu hal berikut:
  - Siram pesan ketika Anda mencetak.

    ```
    import sys

    print('Hello, error!', file=sys.stderr, flush=True)
    ```

  - Siram pesan setelah Anda mencetak. Anda dapat mengirim beberapa pesan sebelum Anda menyiram stream.

    ```
    import sys

    print('Hello, error!', file=sys.stderr)
    sys.stderr.flush()
    ```

Untuk informasi selengkapnya tentang cara memverifikasi bahwa skrip Python mengeluarkan pesan log, lihat Memantau AWS IoT Greengrass log.

Konfigurasi komponen tidak diperbarui saat mengubah konfigurasi default

Saat Anda mengubah `DefaultConfiguration` resep komponen, konfigurasi default baru tidak akan menggantikan konfigurasi komponen yang ada selama penerapan. Untuk menerapkan konfigurasi default baru, Anda harus mengatur ulang konfigurasi komponen ke pengaturan defaultnya. Saat Anda menerapkan komponen, tentukan satu string kosong sebagai pembaruan reset.

Console

Setel ulang jalur

```
[""]
```

AWS CLI

Perintah berikut membuat penyebaran ke perangkat inti.

```
aws greengrassv2 create-deployment --cli-input-json file://reset-configuration-
deployment.json
```

`reset-configuration-deployment.json`File berisi dokumen JSON berikut.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.HelloWorld": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {,
        "reset": [""]
      }
    }
  }
}
```

Greengrass CLI

Perintah Greengrass CLI berikut membuat penerapan lokal pada perangkat inti.

```
sudo greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.HelloWorld=1.0.0" \
  --update-config reset-configuration-deployment.json
```

`reset-configuration-deployment.json`File berisi dokumen JSON berikut.

```
{
  "com.example.HelloWorld": {
    "RESET": [""]
  }
}
```

awsiot.greengrasscoreipc.model.UnauthorizedError

Anda mungkin melihat kesalahan ini di log komponen Greengrass ketika komponen tidak memiliki izin untuk melakukan operasi IPC pada sumber daya. Untuk memberikan izin komponen untuk memanggil operasi IPC, tentukan kebijakan otorisasi IPC dalam konfigurasi komponen. Untuk informasi selengkapnya, lihat Otorisasi komponen untuk melakukan operasi IPC.

> **Tip**
>
> Jika Anda mengubah resep `DefaultConfiguration` dalam komponen, Anda harus mengatur ulang konfigurasi komponen ke konfigurasi default yang baru. Saat Anda menerapkan komponen, tentukan satu string kosong sebagai pembaruan reset. Untuk informasi selengkapnya, lihat Konfigurasi komponen tidak diperbarui saat mengubah konfigurasi default.

com.aws.greengrass.authorization.exceptions.AuthorizationException: Duplicate policy ID "<id>" for principal "<componentList>"

Anda mungkin melihat kesalahan ini jika beberapa kebijakan otorisasi IPC, termasuk di semua komponen pada perangkat inti, menggunakan ID kebijakan yang sama.

Periksa kebijakan otorisasi IPC komponen Anda, perbaiki duplikat apa pun, dan coba lagi. Untuk membuat ID kebijakan unik, sebaiknya Anda menggabungkan nama komponen, nama layanan IPC, dan penghitung. Untuk informasi selengkapnya, lihat Otorisasi komponen untuk melakukan operasi IPC.

> **Tip**
>
> Jika Anda mengubah resep `DefaultConfiguration` dalam komponen, Anda harus mengatur ulang konfigurasi komponen ke konfigurasi default yang baru. Saat Anda menerapkan komponen, tentukan satu string kosong sebagai pembaruan reset. Untuk informasi selengkapnya, lihat Konfigurasi komponen tidak diperbarui saat mengubah konfigurasi default.

com.aws.greengrass.tes.CredentialRequestHandler: Error in retrieving AwsCredentials from TES (HTTP 400)

Anda mungkin melihat kesalahan ini ketika perangkat inti tidak bisa mendapatkan AWS kredensil dari layanan pertukaran token. Kode status HTTP 400 menunjukkan bahwa kesalahan ini terjadi karena peran IAM pertukaran token perangkat inti tidak ada atau tidak memiliki hubungan kepercayaan yang memungkinkan penyedia AWS IoT kredensional untuk mengasumsikan itu.

Lakukan hal-hal berikut:

1. Identifikasi peran pertukaran token yang digunakan perangkat inti. Pesan kesalahan mencakup alias AWS IoT peran perangkat inti, yang menunjuk ke peran pertukaran token. Jalankan perintah berikut di komputer pengembangan Anda, dan ganti *`MyGreengrassCoreTokenExchangeRoleAlias`*dengan nama alias AWS IoT peran dari pesan kesalahan.

   ```
   aws iot describe-role-alias --role-alias MyGreengrassCoreTokenExchangeRoleAlias
   ```

   Tanggapan tersebut mencakup Nama Sumber Daya Amazon (ARN) dari peran IAM pertukaran token.

   ```
   {
     "roleAliasDescription": {
       "roleAlias": "MyGreengrassCoreTokenExchangeRoleAlias",
   ```

```
    "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
MyGreengrassCoreTokenExchangeRoleAlias",
    "roleArn": "arn:aws:iam::123456789012:role/MyGreengrassV2TokenExchangeRole",
    "owner": "123456789012",
    "credentialDurationSeconds": 3600,
    "creationDate": "2021-02-05T16:46:18.042000-08:00",
    "lastModifiedDate": "2021-02-05T16:46:18.042000-08:00"
  }
}
```

2. Periksa apakah peran itu ada. Jalankan perintah berikut, dan ganti *MyGreengrassV2TokenExchangeRole* dengan nama peran pertukaran token.

```
aws iam get-role --role-name MyGreengrassV2TokenExchangeRole
```

   Jika perintah mengembalikan `NoSuchEntity` kesalahan, peran tidak ada, dan Anda harus membuatnya. Untuk informasi selengkapnya tentang cara membuat dan mengonfigurasi peran ini, lihatOtorisasi perangkat inti untuk berinteraksi dengan AWS layanan.

3. Periksa apakah peran tersebut memiliki hubungan kepercayaan yang memungkinkan penyedia AWS IoT kredensional untuk mengasumsikannya. Tanggapan dari langkah sebelumnya berisi`AssumeRolePolicyDocument`, yang mendefinisikan hubungan kepercayaan peran. Peran harus mendefinisikan hubungan kepercayaan yang memungkinkan `credentials.iot.amazonaws.com` untuk mengasumsikan itu. Dokumen ini akan terlihat mirip dengan contoh berikut.

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

   Jika hubungan kepercayaan peran tidak memungkinkan `credentials.iot.amazonaws.com` untuk mengasumsikan itu, Anda harus menambahkan hubungan kepercayaan ini ke peran

tersebut. Untuk informasi selengkapnya, lihat Memodifikasi peran dalam Panduan AWS Identity and Access Management Pengguna.

com.aws.greengrass.tes.CredentialRequestHandler: Error in retrieving AwsCredentials from TES (HTTP 403)

Anda mungkin melihat kesalahan ini ketika perangkat inti tidak bisa mendapatkan AWS kredensil dari layanan pertukaran token. Kode status HTTP 403 menunjukkan bahwa kesalahan ini terjadi karena AWS IoT kebijakan perangkat inti tidak memberikan `iot:AssumeRoleWithCertificate` izin untuk alias AWS IoT peran perangkat inti.

Tinjau AWS IoT kebijakan perangkat inti, dan tambahkan `iot:AssumeRoleWithCertificate` izin untuk alias AWS IoT peran perangkat inti. Pesan kesalahan menyertakan alias AWS IoT peran perangkat inti saat ini. Untuk informasi selengkapnya tentang izin ini dan cara memperbarui AWS IoT kebijakan perangkat inti, lihat Kebijakan AWS IoT minimal untuk perangkat inti AWS IoT Greengrass V2 danMemperbarui AWS IoT kebijakan perangkat inti.

com.aws.greengrass.tes.CredentialsProviderError: Could not load credentials from any providers

Anda mungkin melihat kesalahan ini ketika komponen mencoba meminta AWS kredensional dan tidak dapat terhubung ke layanan pertukaran token.

Lakukan hal-hal berikut:

- Periksa apakah komponen mendeklarasikan ketergantungan pada komponen layanan pertukaran token,. `aws.greengrass.TokenExchangeService` Jika tidak, tambahkan ketergantungan dan gunakan kembali komponen.
- Jika komponen berjalan di docker, pastikan Anda menerapkan pengaturan jaringan dan variabel lingkungan yang tepat, sesuai dengan. Gunakan AWS kredensil dalam komponen wadah Docker (Linux)
- Jika komponen ditulis dalam NodeJS, atur dns. setDefaultResultPerintah untuk**`ipv4first`**.
- Periksa `/etc/hosts` entri yang dimulai dengan `::1` dan berisi`localhost`. Hapus entri untuk melihat apakah itu menyebabkan komponen terhubung ke layanan pertukaran token di alamat yang salah.

Received error when attempting to retrieve ECS metadata: Could not connect to the endpoint URL: "<tokenExchangeServiceEndpoint>"

Anda mungkin melihat kesalahan ini ketika komponen tidak menjalankan layanan pertukaran token dan komponen mencoba meminta AWS kredensil.

Lakukan hal-hal berikut:

- Periksa apakah komponen mendeklarasikan ketergantungan pada komponen layanan pertukaran token,. `aws.greengrass.TokenExchangeService` Jika tidak, tambahkan ketergantungan dan gunakan kembali komponen.
- Periksa apakah komponen menggunakan AWS kredensional dalam siklus hidupnya`install`. AWS IoT Greengrass tidak menjamin ketersediaan layanan pertukaran token selama `install` siklus hidup. Perbarui komponen untuk memindahkan kode yang menggunakan AWS kredensil ke dalam `run` siklus hidup `startup` atau, lalu gunakan kembali komponen.

copyFrom: <configurationPath> is already a container, not a leaf

Anda mungkin melihat kesalahan ini ketika Anda mengubah nilai konfigurasi dari tipe kontainer (daftar atau objek) ke tipe non-kontainer (string, angka, atau Boolean). Lakukan hal-hal berikut:

1. Periksa resep komponen untuk melihat apakah konfigurasi defaultnya menetapkan nilai konfigurasi itu ke daftar atau objek. Jika demikian, hapus atau ubah nilai konfigurasi itu.
2. Buat penerapan untuk mengatur ulang nilai konfigurasi itu ke nilai defaultnya. Lihat informasi yang lebih lengkap di Buat deployment dan Perbarui konfigurasi komponen.

Kemudian, Anda dapat mengatur nilai konfigurasi itu ke string, angka, atau Boolean.

com.aws.greengrass.componentmanager.plugins.docker.exceptions.DockerLoginError logging into the registry using credentials - 'The stub received bad data.'

Anda mungkin melihat kesalahan ini di log inti Greengrass saat komponen pengelola aplikasi Docker mencoba mengunduh image Docker dari repositori pribadi di Amazon Elastic Container Registry (Amazon ECR). Kesalahan ini terjadi jika Anda menggunakan `wincred` Docker credential helper (). `docker-credential-wincred` Akibatnya, Amazon ECR tidak dapat menyimpan kredensil login.

Ambil salah satu tindakan berikut:

- Jika Anda tidak menggunakan pembantu kredenal `wincred` Docker, hapus `docker-credential-wincred` program dari perangkat inti.
- Jika Anda menggunakan pembantu kredenal `wincred` Docker, lakukan hal berikut:
  1. Ganti nama `docker-credential-wincred` program pada perangkat inti. Ganti `wincred` dengan nama baru untuk pembantu kredenal Windows Docker. Misalnya, Anda dapat mengganti namanya menjadi`docker-credential-wincredreal`.
  2. Perbarui `credsStore` opsi di file konfigurasi Docker (`.docker/config.json`) untuk menggunakan nama baru untuk pembantu kredensi Windows Docker. Misalnya, jika Anda mengganti nama program menjadi`docker-credential-wincredreal`, perbarui `credsStore` opsi ke`wincredreal`.

     ```
     {
       "credsStore": "wincredreal"
     }
     ```

java.io.IOException: Cannot run program "cmd" ...: [LogonUser] The password for this account has expired.

Anda mungkin melihat kesalahan ini pada perangkat inti Windows ketika pengguna sistem yang menjalankan proses komponen, sepertiggc_user, memiliki kata sandi yang kedaluwarsa. Akibatnya, perangkat lunak AWS IoT Greengrass Core tidak dapat menjalankan proses komponen sebagai pengguna sistem tersebut.

Untuk memperbarui kata sandi pengguna sistem Greengrass

1. Jalankan perintah berikut sebagai administrator untuk mengatur kata sandi pengguna. *Ganti ggc_user dengan pengguna* sistem, dan ganti *kata sandi dengan kata sandi* yang akan disetel.

   ```
   net user ggc_user password
   ```

2. Gunakan PsExec utilitas untuk menyimpan kata sandi baru pengguna di instance Credential Manager untuk LocalSystem akun tersebut. Ganti *kata sandi* dengan kata sandi pengguna yang Anda tetapkan.

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

Tip

Bergantung pada konfigurasi Windows Anda, kata sandi pengguna mungkin diatur untuk kedaluwarsa pada tanggal di masa mendatang. Untuk memastikan aplikasi Greengrass Anda terus beroperasi, lacak kapan kata sandi kedaluwarsa, dan perbarui sebelum kedaluwarsa. Anda juga dapat mengatur kata sandi pengguna agar tidak pernah kedaluwarsa.

- Untuk memeriksa kapan pengguna dan kata sandinya kedaluwarsa, jalankan perintah berikut.

  ```
  net user ggc_user | findstr /C:expires
  ```

- Untuk mengatur kata sandi pengguna agar tidak pernah kedaluwarsa, jalankan perintah berikut.

  ```
  wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
  ```

- Jika Anda menggunakan Windows 10 atau yang lebih baru di mana wmicperintah tidak digunakan lagi, jalankan perintah berikut. PowerShell

  ```
  Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
   'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
  ```

aws.greengrass.StreamManager: Instant exceeds minimum or maximum instant

Saat Anda memutakhirkan manajer aliran v2.0.7 ke versi antara v2.0.8 dan v2.0.11, Anda mungkin melihat kesalahan berikut di log komponen pengelola aliran jika komponen 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 menerapkan manajer aliran v2.0.7 dan Anda ingin meningkatkan ke versi yang lebih baru, Anda harus meningkatkan ke manajer streaming v2.0.12 secara langsung. Untuk informasi selengkapnya tentang komponen pengelola aliran, lihatManajer pengaliran.

Masalah komponen fungsi Lambda perangkat inti

Memecahkan masalah komponen fungsi Lambda pada perangkat inti.

Topik

- The following cgroup subsystems are not mounted: devices, memory
- ipc_client.py:64,HTTP Error 400:Bad Request, b'No subscription exists for the source <label-or-lambda-arn> and subject <label-or-lambda-arn>

The following cgroup subsystems are not mounted: devices, memory

Anda mungkin melihat kesalahan ini saat menjalankan fungsi Lambda dalam kontainer dalam kasus berikut:

- Perangkat inti tidak mengaktifkan cgroup v1 untuk memori atau cgroup perangkat.
- Perangkat inti telah mengaktifkan cgroups v2. Fungsi Greengrass Lambda membutuhkan cgroups v1, dan cgroups v1 dan v2 saling eksklusif.

Untuk mengaktifkan cgroups v1, boot perangkat dengan parameter kernel Linux berikut.

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

> **Tip**
>
> Pada Raspberry Pi, edit `/boot/cmdline.txt` file untuk mengatur parameter kernel perangkat.

ipc_client.py:64,HTTP Error 400:Bad Request, b'No subscription exists for the source <label-or-lambda-arn> and subject <label-or-lambda-arn>

Anda mungkin melihat kesalahan ini saat menjalankan fungsi Lambda V1, yang menggunakan AWS IoT Greengrass Core SDK, pada perangkat inti V2 tanpa menentukan langganan di komponen router langganan lama. Untuk memperbaiki masalah ini, gunakan dan konfigurasikan router langganan lama untuk menentukan langganan yang diperlukan. Untuk informasi selengkapnya, lihat Impor fungsi Lambda V1.

Versi komponen dihentikan

Anda mungkin melihat pemberitahuan di Personal Health Dashboard (PHD) ketika versi komponen pada perangkat inti Anda dihentikan. Versi komponen mengirimkan pemberitahuan ini ke PHD Anda dalam waktu 60 menit setelah dihentikan.

Untuk melihat penerapan mana yang perlu Anda revisi, lakukan hal berikut menggunakan: AWS Command Line Interface

1. Jalankan perintah berikut untuk mendapatkan daftar perangkat inti Anda.

   ```
   aws greengrassv2 list-core-devices
   ```

2. Jalankan perintah berikut untuk mengambil status komponen pada setiap perangkat inti dari Langkah 1. Ganti *coreDeviceName* dengan nama setiap perangkat inti untuk kueri.

   ```
   aws greengrassv2 list-installed-components --core-device-thing-name coreDeviceName
   ```

3. Kumpulkan perangkat inti dengan versi komponen yang dihentikan yang diinstal dari langkah sebelumnya.
4. Jalankan perintah berikut untuk mengambil status semua pekerjaan penerapan untuk setiap perangkat inti dari Langkah 3. Ganti *coreDeviceName* dengan nama perangkat inti untuk kueri.

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

   Tanggapan berisi daftar tugas deployment untuk perangkat inti. Anda dapat merevisi penerapan untuk memilih versi komponen lain. Untuk informasi selengkapnya tentang cara merevisi penerapan, lihat Merevisi penerapan.

Masalah Antarmuka Baris Perintah Greengrass

Memecahkan masalah dengan CLI Greengrass.

Topik

- java.lang.RuntimeException: Unable to create ipc client

java.lang.RuntimeException: Unable to create ipc client

Anda mungkin melihat kesalahan ini saat menjalankan perintah Greengrass CLI dan Anda menentukan folder root yang berbeda dari tempat perangkat lunak Core diinstal. AWS IoT Greengrass

Lakukan salah satu hal berikut untuk mengatur jalur root, dan ganti */greengrass/v2* dengan jalur ke instalasi perangkat lunak AWS IoT Greengrass Core Anda:

- Atur `GGC_ROOT_PATH` variabel lingkungan ke */greengrass/v2*.
- Tambahkan `--ggcRootPath` */greengrass/v2* argumen ke perintah Anda seperti yang ditunjukkan pada contoh berikut.

```
greengrass-cli --ggcRootPath /greengrass/v2 <command> <subcommand> [arguments]
```

AWS Command Line Interface masalah

Memecahkan AWS CLI masalah untuk. AWS IoT Greengrass V2

Topik

- Error: Invalid choice: 'greengrassv2'

Error: Invalid choice: 'greengrassv2'

Anda mungkin melihat kesalahan ini ketika Anda menjalankan AWS IoT Greengrass V2 perintah dengan AWS CLI (misalnya,`aws greengrassv2 list-core-devices`).

Kesalahan ini menunjukkan bahwa Anda memiliki versi AWS CLI yang tidak mendukung AWS IoT Greengrass V2. Untuk menggunakannya AWS IoT Greengrass V2 AWS CLI, Anda harus memiliki salah satu versi berikut atau yang lebih baru:

- Versi AWS CLI V1 minimum: v1.18.197
- Versi AWS CLI V2 minimum: v2.1.11

> **Tip**
>
> Anda dapat menjalankan perintah berikut untuk memeriksa versi AWS CLI yang Anda miliki.
>
> ```
> aws --version
> ```

Untuk mengatasi masalah ini, perbarui AWS CLI ke versi yang lebih baru yang mendukung AWS IoT Greengrass V2. Untuk informasi selengkapnya, lihat Menginstal, memperbarui, dan mencopot instalasi AWS CLI di Panduan Pengguna AWS Command Line Interface .

Kode kesalahan penyebaran terperinci

Gunakan kode kesalahan dan solusi di bagian ini untuk membantu menyelesaikan masalah dengan penyebaran komponen saat menggunakan Greengrass nucleus versi 2.8.0 atau yang lebih baru.

Inti Greengrass melaporkan kesalahan penyebaran sebagai hierarki dari yang paling tidak spesifik hingga kode paling spesifik yang tersedia. Anda dapat menggunakan hierarki ini untuk membantu menentukan alasan kesalahan penyebaran. Misalnya, berikut ini adalah hirarki kesalahan yang mungkin:

- DEPLOYMENT_FAILURE
  - ARTIFACT_DOWNLOAD_ERROR
    - IO_ERROR
      - DISK_SPACE_CRITICAL

Kode kesalahan diatur ke dalam jenis. Setiap jenis mewakili kelas kesalahan yang dapat terjadi.AWS IoT Greengrassmelaporkan jenis kesalahan ini di konsol, API, danAWS CLI. Mungkin ada lebih dari satu jenis kesalahan, tergantung pada kesalahan yang dilaporkan dalam hirarki kesalahan. Untuk contoh sebelumnya, jenis kesalahan yang dikembalikan adalah`DEVICE_ERROR`.

Jenisnya adalah:

- PERMISSION_ERROR- Akses ke operasi yang membutuhkan izin ditolak.

- REQUEST_ERROR- Terjadi kesalahan karena masalah dalam dokumen penyebaran.
- COMPONENT_RECIPE_ERROR- Terjadi kesalahan karena masalah dalam resep komponen.
- AWS_COMPONENT_ERROR- Terjadi kesalahan saat memulai atau menghapusAWSkomponen yang disediakan.
- USER_COMPONENT_ERROR- Terjadi kesalahan saat memulai atau menghapus komponen pengguna.
- COMPONENT_ERROR- Terjadi kesalahan saat memulai atau menghapus komponen, tetapi inti Greengrass tidak dapat menentukan apakah komponennya adalahAWSkomponen yang disediakan atau komponen pengguna.
- DEVICE_ERROR- Terjadi kesalahan dengan I/O lokal atau kesalahan perangkat lain.
- DEPENDENCY_ERROR- Penerapan gagal mengunduh artefak dari Amazon S3 atau menarik gambar dari registri ECR.
- HTTP_ERROR- Terjadi kesalahan dengan permintaan HTTP.
- NETWORK_ERROR- Terjadi kesalahan dengan jaringan perangkat.
- NUCLEUS_ERROR- Inti Greengrass tidak dapat menemukan komponen atau tidak dapat menemukan versi inti aktif.
- SERVER_ERROR- Server mengembalikan kesalahan 500 sebagai tanggapan atas permintaan.
- CLOUD_SERVICE_ERROR- Terjadi kesalahan denganAWS IoT Greengrasslayanan cloud.
- TIDAK DIKETAHUI_ERROR- Pengecualian dicentang dilemparkan oleh komponen.

Banyak kesalahan di bagian ini melaporkan informasi tambahan diAWS IoT GreengrassLog inti. Log ini disimpan pada sistem file lokal perangkat inti. Ada log untukAWS IoT GreengrassPerangkat lunak inti inti dan untuk setiap komponen individu. Untuk informasi tentang mengakses log, lihatAkses log sistem file.

Kesalahan izin

ACCESS_DITOLAK

Anda mungkin mendapatkan kesalahan ini saatAWSoperasi layanan mengembalikan kesalahan 403 karena izin tidak diatur dengan benar. Periksa kode kesalahan yang lebih spesifik untuk detailnya.

GET_DEPLOYMENT_CONFIGURATION_ACCESS_DITOLAK

Anda mungkin mendapatkan kesalahan ini saatAWS IoTkebijakan tidak mengizinkan izin untuk memanggil`GetDeploymentConfiguration`operasi. Tambahkan`greengrass::GetDeploymentConfiguration`izin untuk kebijakan perangkat inti.

GET_COMPONENT_VERSION_ARTIFACT_ACCESS_DITOLAK

Anda mungkin mendapatkan kesalahan ini saat perangkat intiAWS IoTkebijakan tidak mengizinkan`greengrass:GetComponentVersionArtifact`izin. Tambahkan izin ke kebijakan perangkat inti.

RESOLVE_COMPONENT_CANDIDATES_ACCESS_DITOLAK

Anda mungkin mendapatkan kesalahan ini saat perangkat intiAWS IoTkebijakan tidak mengizinkan`greengrass:ResolveComponentCandidates`izin. Tambahkan izin ke kebijakan perangkat inti.

GET_ECR_CREDENTIAL_ERROR

Anda mungkin mendapatkan kesalahan ini ketika penyebaran tidak dapat mengotentikasi dengan registri pribadi di ECR. Periksa log untuk kesalahan tertentu dan kemudian coba penyebaran lagi.

USER_NOT_AUTHORIZED_FOR_DOCKER

Anda mungkin mendapatkan kesalahan ini ketika pengguna Greengrass tidak diizinkan untuk menggunakan Docker. Pastikan bahwa Anda menjalankan Greengrass sebagai root atau bahwa pengguna ditambahkan ke`docker`kelompok. Kemudian coba penyebarannya lagi.

S3_ACCESS_DITOLAK

Anda mungkin mendapatkan kesalahan ini saat operasi Amazon S3 mengembalikan kesalahan 403. Periksa kode kesalahan atau log tambahan untuk detailnya.

S3_HEAD_OBJECT_ACCESS_DITOLAK

Anda mungkin mendapatkan kesalahan ini baik ketika peran pertukaran token perangkat tidak mengizinkanAWS IoT GreengrassPerangkat lunak inti untuk mengunduh artefak komponen dari URL objek S3 yang Anda tentukan dalam resep komponen atau bahwa artefak komponen tidak tersedia. Periksa apakah peran pertukaran token memungkinkan`s3:GetObject`untuk URL objek S3 di mana artefak tersedia dan artefak hadir.

S3_GET_BUCKET_LOCATION_ACCESS_DITOLAK

Anda mungkin mendapatkan kesalahan ini saat peran pertukaran token perangkat tidak mengizinkan`s3:GetBucketLocation`izin untuk bucket Amazon S3 tempat artefak tersedia. Periksa apakah perangkat mengizinkan izin, lalu coba penyebarannya lagi.

S3_GET_OBJECT_ACCESS_DITOLAK

Anda mungkin mendapatkan kesalahan ini baik ketika peran pertukaran token perangkat tidak mengizinkanAWS IoT GreengrassPerangkat lunak inti untuk mengunduh artefak komponen dari URL objek S3 yang Anda tentukan dalam resep komponen atau bahwa artefak komponen tidak tersedia. Periksa apakah peran pertukaran token memungkinkan`s3:GetObject`untuk URL objek S3 di mana artefak tersedia dan artefak hadir.

Kesalahan permintaan

NUCLEUS_MISSING_REQUIRED_ABILITIES

Anda mungkin mendapatkan kesalahan ini ketika versi nucleus dalam penyebaran tidak mampu operasi yang diminta, seperti mengunduh konfigurasi besar atau menyetel batas sumber daya Linux. Coba lagi penyebaran dengan versi nukleus yang mendukung operasi.

MULTIPLE_NUCLEUS_RESOLVED_ERROR

Anda mungkin mendapatkan kesalahan ini ketika penyebaran mencoba untuk menyebarkan beberapa komponen nukleus. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

COMPONENT_CIRCULAR_DEPENDENCY_ERROR

Anda mungkin mendapatkan kesalahan ini ketika dua komponen dalam penyebaran Anda bergantung satu sama lain. Merevisi pengaturan komponen sehingga komponen dalam penyebaran Anda tidak bergantung satu sama lain.

UNAUTHORIZED_NUCLEUS_MINOR_VERSION_UPDATE

Anda mungkin mendapatkan kesalahan ini ketika komponen dalam penyebaran Anda memerlukan pembaruan versi minor inti, tetapi versi tersebut tidak ditentukan dalam penyebaran. Ini membantu mengurangi pembaruan versi minor yang tidak disengaja untuk komponen yang bergantung pada versi yang berbeda. Sertakan versi nukleus minor baru dalam penyebaran.

HILANG_DOCKER_APPLICATION_MANAGER

Anda mungkin mendapatkan kesalahan ini saat Anda menerapkan komponen Docker tanpa menerapkan manajer aplikasi Docker. Pastikan bahwa penerapan Anda menyertakan manajer aplikasi Docker.

HILANG_TOKEN_EXCHANGE_SERVICE

Anda mungkin mendapatkan kesalahan ini ketika penyebaran ingin mengunduh artefak gambar Docker dari registri ECR pribadi tanpa menerapkan layanan pertukaran token. Pastikan bahwa penyebaran Anda mencakup layanan pertukaran token.

COMPONENT_VERSION_REQUIREMENTS_NOT_MET

Anda mungkin mendapatkan kesalahan ini ketika ada konflik kendala versi atau versi komponen tidak ada. Untuk informasi selengkapnya, lihat Error: com.aws.greengrass.componentmanager.exceptions.NoAvailableComponentVersionException: Failed to negotiate component <name> version with cloud and no local applicable version satisfying requirement <requirements>.

THROTTLING_ERROR

Anda mungkin mendapatkan kesalahan ini saatAWSoperasi layanan melebihi kuota tarif. Coba lagi deployment.

CONFLICTED_REQUEST

Anda mungkin mendapatkan kesalahan ini saatAWSoperasi layanan mengembalikan kesalahan 409 karena penyebaran Anda mencoba untuk melakukan lebih dari satu operasi pada satu waktu. Coba lagi deployment.

RESOURCE_NOT_FOUND

Anda mungkin mendapatkan kesalahan ini saatAWSoperasi layanan mengembalikan kesalahan 404 karena sumber daya tidak dapat ditemukan. Periksa log untuk sumber daya yang hilang.

RUN_WITH_CONFIG_NOT_VALID

Anda mungkin mendapatkan kesalahan ini saat`posixUser`,`posixGroup`, atau`windowsUser`informasi yang ditentukan untuk menjalankan komponen tidak valid. Periksa apakah pengguna valid dan kemudian coba lagi penyebaran.

UNSUPPORTED_REGION

Anda mungkin mendapatkan kesalahan ini saat Wilayah yang ditentukan untuk penyebaran tidak didukung olehAWS IoT Greengrass. Periksa Wilayah dan coba penyebarannya lagi.

IOT_CRED_ENDPOINT_NOT_VALID

Anda mungkin mendapatkan kesalahan ini saatAWS IoTtitik akhir kredensi yang ditentukan dalam konfigurasi tidak valid. Periksa titik akhir dan coba permintaan Anda lagi.

IOT_DATA_ENDPOINT_NOT_VALID

Anda mungkin mendapatkan kesalahan ini saatAWS IoTendpoint data yang ditentukan dalam konfigurasi tidak valid. Periksa titik akhir dan coba permintaan Anda lagi.

S3_HEAD_OBJECT_RESOURCE_NOT_FOUND

Anda mungkin mendapatkan kesalahan ini ketika artefak komponen tidak tersedia di URL objek S3 yang Anda tentukan dalam resep komponen. Periksa apakah Anda mengunggah artefak ke bucket S3 dan bahwa URI artefak cocok dengan URL objek S3 artefak di bucket.

S3_GET_BUCKET_LOCATION_RESOURCE_NOT_FOUND

Anda mungkin mendapatkan kesalahan ini saat bucket Amazon S3 tidak ditemukan. Periksa apakah bucket ada dan coba penyebarannya lagi.

S3_GET_OBJECT_RESOURCE_NOT_FOUND

Anda mungkin mendapatkan kesalahan ini ketika artefak komponen tidak tersedia di URL objek S3 yang Anda tentukan dalam resep komponen. Periksa apakah Anda mengunggah artefak ke bucket S3 dan bahwa URI artefak cocok dengan URL objek S3 artefak di bucket.

IO_MAPPING_ERROR

Anda mungkin mendapatkan kesalahan ini ketika kesalahan I/O terjadi ketika parsing dokumen penyebaran atau resep. Periksa kode kesalahan atau log tambahan untuk detailnya.

Kesalahan resep komponen

RECIPE_PARSE_ERROR

Anda mungkin mendapatkan kesalahan ini ketika resep penyebaran tidak dapat diurai karena ada kesalahan dalam struktur resep. Periksa apakah resep diformat dengan benar dan coba penyebarannya lagi.

RECIPE_METADATA_PARSE_ERROR

Anda mungkin mendapatkan kesalahan ini saat metadata resep penyebaran yang diunduh dari cloud tidak dapat diurai. Hubungi AWS Support.

ARTIFACT_URI_NOT_VALID

Anda mungkin mendapatkan kesalahan ini ketika URI artefak dalam resep tidak diformat dengan benar. Periksa log untuk URI yang tidak valid, perbarui URI dalam resep, lalu coba penyebaran lagi.

S3_ARTIFACT_URI_NOT_VALID

Anda mungkin mendapatkan kesalahan ini ketika URI Amazon S3 artefak dalam resep tidak valid. Periksa log untuk URI yang tidak valid, perbarui URI dalam resep, lalu coba penyebaran lagi.

DOCKER_ARTIFACT_URI_NOT_VALID

Anda mungkin mendapatkan kesalahan ini ketika URI Docker artefak dalam resep tidak valid. Periksa log untuk URI yang tidak valid, perbarui URI dalam resep, lalu coba penyebaran lagi.

EMPTY_ARTIFACT_URI

Anda mungkin mendapatkan kesalahan ini ketika URI artefak tidak ditentukan dalam resep. Periksa log untuk artefak yang kehilangan URI, perbarui URI dalam resep, lalu coba penyebaran lagi.

EMPTY_ARTIFACT_SCHEME

Anda mungkin mendapatkan kesalahan ini ketika skema URI tidak didefinisikan untuk artefak. Periksa log untuk URI yang tidak valid, perbarui URI dalam resep, lalu coba penyebaran lagi.

UNSUPPORTED_ARTIFACT_SCHEME

Anda mungkin mendapatkan kesalahan ini saat skema URI tidak didukung oleh versi nucleus yang sedang berjalan. URI tidak valid atau Anda perlu memperbarui versi nucleus. Jika URI tidak valid, periksa log untuk URI yang tidak valid, perbarui URI dalam resep, lalu coba penyebaran lagi.

RECIPE_MISSING_MANIFEST

Anda mungkin mendapatkan kesalahan ini saat bagian manifes tidak disertakan dalam resep. Tambahkan manifes ke resep dan coba penyebarannya lagi.

RECIPE_MISSING_ARTIFACT_HASH_ALGORITHM

Anda mungkin mendapatkan kesalahan ini ketika artefak yang tidak lokal ditentukan dalam resep tanpa algoritma hash. Tambahkan algoritma ke artefak dan kemudian coba permintaan lagi.

ARTIFACT_CHECKSUM_TIDAK COCOK

Anda mungkin mendapatkan kesalahan ini ketika artefak yang diunduh memiliki intisari yang berbeda dari yang ditentukan dalam resep. Pastikan bahwa resep berisi intisari yang benar dan kemudian coba penyebaran lagi. Untuk informasi selengkapnya, lihat Error: com.aws.greengrass.componentmanager.exceptions.ArtifactChecksumMismatchException: Integrity check for downloaded artifact failed. Probably due to file corruption..

COMPONENT_DEPENDENCY_NOT_VALID

Anda mungkin mendapatkan kesalahan ini ketika jenis dependensi yang ditentukan dalam resep penyebaran tidak valid. Periksa resep dan kemudian coba permintaan Anda lagi.

CONFIG_INTERPOLATE_ERROR

Anda mungkin mendapatkan kesalahan ini saat menginterpolasi variabel resep. Periksa log untuk detailnya.

IO_MAPPING_ERROR

Anda mungkin mendapatkan kesalahan ini ketika kesalahan I/O terjadi ketika parsing dokumen penyebaran atau resep. Periksa kode kesalahan atau log tambahan untuk detailnya.

AWSkesalahan komponen, kesalahan komponen pengguna, kesalahan komponen

Kode kesalahan berikut dikembalikan ketika ada masalah dengan komponen. Jenis kesalahan aktual yang dilaporkan tergantung pada komponen tertentu yang menimbulkan kesalahan. Jika inti Greengrass mengidentifikasi komponen sebagai salah satu yang disediakan olehAWS IoT Greengrass, ia kembaliAWS_COMPONENT_ERROR. Jika komponen diidentifikasi sebagai komponen pengguna, inti Greengrass kembaliUSER_COMPONENT_ERROR. Jika inti Greengrass tidak tahu, ia kembaliCOMPONENT_ERROR.

COMPONENT_UPDATE_ERROR

Anda mungkin mendapatkan kesalahan ini ketika komponen tidak diperbarui selama penyebaran. Periksa kode kesalahan tambahan atau periksa log untuk melihat apa yang menyebabkan kesalahan.

COMPONENT_BROKEN

Anda mungkin mendapatkan kesalahan ini ketika komponen rusak selama penyebaran. Periksa log komponen untuk rincian kesalahan dan kemudian coba penyebaran lagi.

REMOVE_COMPONENT_ERROR

Anda mungkin mendapatkan kesalahan ini ketika nukleus tidak dapat menghapus komponen selama penyebaran. Periksa log untuk rincian kesalahan dan kemudian coba penyebaran lagi.

COMPONENT_BOOTSTRAP_BATAS WAKTU

Anda mungkin mendapatkan kesalahan ini ketika tugas bootstrap komponen memakan waktu lebih lama dari batas waktu yang dikonfigurasi. Tingkatkan batas waktu atau kurangi waktu eksekusi tugas bootstrap, lalu coba penyebarannya lagi.

COMPONENT_BOOTSTRAP_ERROR

Anda mungkin mendapatkan kesalahan ini ketika tugas bootstrap komponen memiliki kesalahan. Periksa log untuk mengetahui detail kesalahan, lalu coba penyebarannya lagi.

COMPONENT_CONFIGURATION_NOT_VALID

Anda mungkin mendapatkan kesalahan ini ketika nukleus tidak dapat memvalidasi konfigurasi yang diterapkan untuk komponen. Periksa log untuk mengetahui detail kesalahan, lalu coba penyebarannya lagi.

Kesalahan perangkat

IO_WRITE_ERROR

Anda mungkin mendapatkan kesalahan ini saat menulis ke file. Periksa log untuk detailnya.

IO_READ_ERROR

Anda mungkin mendapatkan kesalahan ini saat membaca dari file. Periksa log untuk detailnya.

DISK_SPACE_CRITICAL

Anda mungkin mendapatkan kesalahan ini ketika tidak ada cukup ruang disk untuk menyelesaikan permintaan penyebaran. Anda harus memiliki setidaknya 20 Mb ruang yang tersedia, atau cukup untuk memegang artefak yang lebih besar. Kosongkan beberapa ruang disk dan kemudian coba lagi penyebaran.

IO_FILE_ATTRIBUTE_ERROR

Anda mungkin mendapatkan kesalahan ini ketika ukuran file yang ada tidak dapat diambil dari sistem file. Periksa log untuk detailnya.

SET_PERMISSION_ERROR

Anda mungkin mendapatkan kesalahan ini ketika izin tidak dapat diatur pada direktori artefak atau artefak yang diunduh. Periksa log untuk detailnya.

IO_UNZIP_ERROR

Anda mungkin mendapatkan kesalahan ini ketika artefak tidak dapat dibuka ritsleting. Periksa log untuk detailnya.

LOCAL_RECIPE_NOT_FOUND

Anda mungkin mendapatkan kesalahan ini ketika salinan lokal dari file resep tidak dapat ditemukan. Coba penyebarannya lagi.

LOCAL_RECIPE_RUSAK

Anda mungkin mendapatkan kesalahan ini ketika salinan lokal resep telah berubah sejak diunduh. Hapus salinan resep yang ada dan coba penyebarannya lagi.

LOCAL_RECIPE_METADATA_NOT_FOUND

Anda mungkin mendapatkan kesalahan ini ketika salinan lokal dari file metadata resep tidak dapat ditemukan. Coba penyebarannya lagi.

LAUNCH_DIRECTORY_RUSAK

Anda mungkin mendapatkan kesalahan ini ketika direktori yang digunakan untuk meluncurkan inti Greengrass (`/greengrass/v2/alts/current`) telah dimodifikasi sejak terakhir kali nukleus dimulai. Mulai ulang nukleus dan kemudian coba lagi penyebaran.

HASHING_ALGORITHM_TIDAK TERSEDIA

Anda mungkin mendapatkan kesalahan ini ketika distribusi Java perangkat tidak mendukung algoritma hashing yang diperlukan atau ketika algoritma hash yang ditentukan dalam resep komponen tidak valid.

DEVICE_CONFIG_NOT_VALID_FOR_ARTIFACT_DOWNLOAD

Anda mungkin mendapatkan kesalahan ini ketika ada kesalahan dalam konfigurasi perangkat yang mencegah penyebaran mengunduh artefak dari Amazon S3 atau cloud Greengrass. Periksa log untuk kesalahan konfigurasi tertentu dan kemudian coba lagi penyebaran.

Kesalahan dependensi

DOCKER_ERROR

Anda mungkin mendapatkan kesalahan ini saat menarik gambar Docker. Periksa kode kesalahan atau log tambahan untuk detailnya.

DOCKER_SERVICE_TIDAK TERSEDIA

Anda mungkin mendapatkan kesalahan ini ketika Greengrass tidak dapat masuk ke registri Docker. Periksa log untuk kesalahan tertentu dan kemudian coba penyebaran lagi.

DOCKER_LOGIN_ERROR

Anda mungkin mendapatkan kesalahan ini saat terjadi kesalahan tak terduga saat masuk ke Docker. Periksa log untuk kesalahan tertentu dan kemudian coba penyebaran lagi.

DOCKER_PULL_ERROR

Anda mungkin mendapatkan kesalahan ini saat terjadi kesalahan tak terduga saat menarik image Docker dari registri. Periksa log untuk kesalahan tertentu dan kemudian coba penyebaran lagi.

DOCKER_IMAGE_NOT_VALID

Anda mungkin mendapatkan kesalahan ini ketika gambar Docker yang diminta tidak ada. Periksa log untuk kesalahan tertentu dan coba penyebarannya lagi.

DOCKER_IMAGE_QUERY_ERROR

Anda mungkin mendapatkan kesalahan ini saat terjadi kegagalan tak terduga saat melakukan query ke Docker untuk image yang tersedia. Periksa log untuk kesalahan tertentu dan coba penyebarannya lagi.

S3_ERROR

Anda mungkin mendapatkan kesalahan ini saat mengunduh artefak Amazon S3. Periksa kode kesalahan atau log tambahan untuk detailnya.

S3_RESOURCE_NOT_FOUND

Anda mungkin mendapatkan kesalahan ini saat operasi Amazon S3 mengembalikan kesalahan 404. Periksa kode kesalahan atau log tambahan untuk detailnya.

S3_BAD_REQUEST

Anda mungkin mendapatkan kesalahan ini saat operasi Amazon S3 mengembalikan kesalahan 400. Periksa log untuk kesalahan tertentu dan coba permintaan lagi.

Kesalahan HTTP

HTTP_REQUEST_ERROR

Anda mungkin mendapatkan kesalahan ini ketika terjadi kesalahan saat membuat permintaan HTTP. Periksa log untuk kesalahan tertentu.

DOWNLOAD_DEPLOYMENT_DOCUMENT_ERROR

Anda mungkin mendapatkan kesalahan ini ketika terjadi kesalahan HTTP saat mengunduh dokumen penyebaran. Periksa log untuk kesalahan HTTP tertentu.

GET_GREENGRASS_ARTIFACT_SIZE_ERROR

Anda mungkin mendapatkan kesalahan ini ketika terjadi kesalahan HTTP saat mendapatkan ukuran artefak komponen publik. Periksa log untuk kesalahan HTTP tertentu.

DOWNLOAD_GREENGRASS_ARTIFACT_ERROR

Anda mungkin mendapatkan kesalahan ini ketika terjadi kesalahan HTTP saat mengunduh artefak komponen publik. Periksa log untuk kesalahan HTTP tertentu.

Kesalahan jaringan

NETWORK_ERROR

Anda mungkin mendapatkan kesalahan ini ketika ada masalah koneksi selama penyebaran. Periksa koneksi perangkat ke Internet dan coba penyebarannya lagi.

Kesalahan nukleus

PERMINTAAN

Anda mungkin mendapatkan kesalahan ini saatAWSoperasi cloud mengembalikan kesalahan 400. Periksa log untuk melihat API mana yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

NUCLEUS_VERSION_NOT_FOUND

Anda mungkin mendapatkan kesalahan ini ketika perangkat inti tidak dapat menemukan versi inti aktif. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman

pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

NUCLEUS_RESTART_FAILURE

Anda mungkin mendapatkan kesalahan ini saat nukleus tidak memulai ulang selama penerapan apa pun yang memerlukan restart nukleus. Periksa log pemuat untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

INSTALLED_COMPONENT_NOT_FOUND

Anda mungkin mendapatkan kesalahan ini ketika nukleus tidak dapat menemukan komponen yang diinstal. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

DEPLOYMENT_DOCUMENT_NOT_VALID

Anda mungkin mendapatkan kesalahan ini saat perangkat menerima dokumen penyebaran yang tidak valid. Periksa kode kesalahan tambahan atau periksa log untuk melihat apa yang menyebabkan kesalahan.

EMPTY_DEPLOYMENT_REQUEST

Anda mungkin mendapatkan kesalahan ini saat perangkat menerima permintaan penyebaran kosong. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

DEPLOYMENT_DOCUMENT_PARSE_ERROR

Anda mungkin mendapatkan kesalahan ini ketika format permintaan penyebaran tidak cocok dengan format yang diharapkan. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

COMPONENT_METADATA_NOT_VALID_IN_DEPLOYMENT

Anda mungkin mendapatkan kesalahan ini saat permintaan penyebaran berisi metadata komponen yang tidak valid. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

LAUNCH_DIRECTORY_RUSAK

Anda mungkin mendapatkan kesalahan ini saat memindahkan perangkat Greengrass dari satu grup ke grup lainnya, dan kemudian kembali ke grup asli dengan penerapan yang memerlukan Greengrass untuk memulai ulang. Untuk mengatasi kesalahan, buat ulang direktori peluncuran untuk Greengrass di perangkat.

Untuk informasi selengkapnya, lihat Error: com.aws.greengrass.deployment.exceptions.DeploymentException: Unable to process deployment. Greengrass launch directory is not set up or Greengrass is not set up as a system service.

Kesalahan server

SERVER_ERROR

Anda mungkin mendapatkan kesalahan ini saatAWSoperasi layanan mengembalikan kesalahan 500 karena layanan tidak dapat memproses permintaan sekarang. Coba lagi penyebaran nanti.

S3_SERVER_ERROR

Anda mungkin mendapatkan kesalahan ini saat operasi Amazon S3 mengembalikan kesalahan 500. Periksa kode kesalahan atau log tambahan untuk detailnya.

Kesalahan layanan cloud

RESOLVE_COMPONENT_CANDIDATES_BAD_RESPONSE

Anda mungkin mendapatkan kesalahan ini saat layanan cloud Greengrass mengirimkan respons yang tidak kompatibel ke`ResolveComponentCandidates`operasi. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

DEPLOYMENT_DOCUMENT_SIZE_EXCEEDED

Anda mungkin mendapatkan kesalahan ini ketika dokumen penyebaran yang diminta melebihi kuota ukuran maksimum. Kurangi ukuran dokumen penyebaran dan coba penyebarannya lagi.

GREENGRASS_ARTIFACT_SIZE_NOT_FOUND

Anda mungkin mendapatkan kesalahan ini ketika Greengrass tidak bisa mendapatkan ukuran artefak komponen publik. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

DEPLOYMENT_DOCUMENT_NOT_VALID

Anda mungkin mendapatkan kesalahan ini saat perangkat menerima dokumen penyebaran yang tidak valid. Periksa kode kesalahan tambahan atau periksa log untuk melihat apa yang menyebabkan kesalahan.

EMPTY_DEPLOYMENT_REQUEST

Anda mungkin mendapatkan kesalahan ini saat perangkat menerima permintaan penyebaran kosong. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

DEPLOYMENT_DOCUMENT_PARSE_ERROR

Anda mungkin mendapatkan kesalahan ini ketika format permintaan penyebaran tidak cocok dengan format yang diharapkan. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

COMPONENT_METADATA_NOT_VALID_IN_DEPLOYMENT

Anda mungkin mendapatkan kesalahan ini saat permintaan penyebaran berisi metadata komponen yang tidak valid. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

Kesalahan generik

Kesalahan generik ini tidak memiliki jenis kesalahan terkait.

DEPLOYMENT_INTERRUPTED

Anda mungkin mendapatkan kesalahan ini ketika penyebaran tidak dapat diselesaikan karena shutdown nukleus atau peristiwa eksternal lainnya. Periksa kode kesalahan atau log tambahan untuk detailnya.

ARTIFACT_DOWNLOAD_ERROR

Anda mungkin mendapatkan kesalahan ini ketika ada masalah mengunduh artefak. Periksa kode kesalahan atau log tambahan untuk detailnya.

NO_AVAILABLE_COMPONENT_VERSION

Anda mungkin mendapatkan kesalahan ini ketika versi komponen tidak ada di cloud atau lokal, atau jika ada konflik resolusi dependensi. Periksa kode kesalahan atau log tambahan untuk detailnya.

COMPONENT_PACKAGE_LOADING_ERROR

Anda mungkin mendapatkan kesalahan ini ketika kesalahan memproses artefak download. Periksa kode kesalahan atau log tambahan untuk detailnya.

CLOUD_API_ERROR

Anda mungkin mendapatkan galat ini ketika terjadi galat memanggilAWSAPI layanan. Periksa kode kesalahan atau log tambahan untuk detailnya.

IO_ERROR

Anda mungkin mendapatkan kesalahan ini ketika kesalahan I/O terjadi selama penyebaran. Periksa kode kesalahan atau log tambahan untuk detailnya.

COMPONENT_UPDATE_ERROR

Anda mungkin mendapatkan kesalahan ini ketika komponen tidak diperbarui selama penyebaran. Periksa kode kesalahan tambahan atau periksa log untuk melihat apa yang menyebabkan kesalahan.

Kesalahan tak dikenal

DEPLOYMENT_FAILURE

Anda mungkin mendapatkan kesalahan ini ketika penyebaran gagal karena pengecualian dicentang dilemparkan. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

DEPLOYMENT_TYPE_NOT_VALID

Anda mungkin mendapatkan kesalahan ini ketika jenis penyebaran tidak valid. Periksa log untuk melihat apa yang menyebabkan kesalahan, lalu periksa halaman pembaruan perangkat lunak

nukleus untuk melihat apakah masalah telah diperbaiki di versi nukleus yang lebih baru, atau kontakAWS Support.

Kode status komponen rinci

Gunakan kode status dan solusi di bagian ini untuk membantu menyelesaikan masalah dengan komponen saat menggunakan Greengrass nucleus versi 2.8.0 atau yang lebih baru.

Banyak status dalam topik ini melaporkan informasi tambahan di log AWS IoT Greengrass Inti. Log ini disimpan pada sistem file lokal perangkat inti. Ada log untuk setiap komponen individu. Untuk informasi tentang mengakses log, lihatAkses log sistem file.

INSTALL_ERROR

Anda mungkin mendapatkan ini ketika terjadi kesalahan saat menjalankan skrip instalasi. Kode kesalahan dilaporkan dalam log komponen. Periksa skrip instalasi untuk kesalahan dan deploy komponen Anda lagi.

INSTALL_CONFIG_NOT_VALID

Anda mungkin mendapatkan kesalahan ini ketika instalasi komponen tidak dapat diselesaikan karena `Install` bagian resep tidak valid. Periksa bagian instalasi resep Anda untuk kesalahan dan coba penyebarannya lagi.

INSTALL_IO_ERROR

Anda mungkin mendapatkan ini ketika kesalahan I/O terjadi selama instalasi komponen. Periksa komponen kesalahan log untuk detail tentang kesalahan.

INSTALL_MISSING_DEFAULT_RUNDENGAN

Anda mungkin mendapatkan kesalahan ini ketika tidak AWS IoT Greengrass dapat menentukan pengguna atau grup yang akan digunakan saat menginstal komponen. Periksa untuk memastikan bahwa `runWith` bagian resep instalasi Anda menyertakan pengguna atau grup yang valid.

INSTALL_TIMEOUT

Anda mungkin mendapatkan kesalahan ini ketika skrip instalasi tidak selesai dalam periode batas waktu yang dikonfigurasi. Baik meningkatkan `Timeout` periode yang ditentukan di `Install` bagian resep atau memodifikasi skrip instalasi Anda untuk menyelesaikan dalam batas waktu yang dikonfigurasi.

STARTUP_ERROR

Anda mungkin mendapatkan ini ketika terjadi kesalahan saat menjalankan skrip startup. Kode kesalahan dilaporkan dalam log komponen. Periksa skrip instalasi untuk kesalahan dan deploy komponen Anda lagi.

STARTUP_CONFIG_NOT_VALID

Anda mungkin mendapatkan kesalahan ini ketika instalasi komponen tidak dapat diselesaikan karena `Startup` bagian resep tidak valid. Periksa bagian startup resep Anda untuk kesalahan dan coba penyebaran lagi.

STARTUP_IO_ERROR

Anda mungkin mendapatkan ini ketika kesalahan I/O terjadi selama startup komponen. Periksa komponen kesalahan log untuk detail tentang kesalahan.

STARTUP_MISSING_DEFAULT_RUNDENGAN

Anda mungkin mendapatkan kesalahan ini saat tidak AWS IoT Greengrass dapat menentukan pengguna atau grup yang akan digunakan saat menjalankan komponen. Periksa untuk memastikan bahwa `runWith` bagian resep startup Anda menyertakan pengguna atau grup yang valid.

STARTUP_BATAS WAKTU

Anda mungkin mendapatkan kesalahan ini ketika skrip startup tidak selesai dalam periode batas waktu yang dikonfigurasi. Baik meningkatkan `Timeout` periode yang ditentukan di `Startup` bagian resep atau memodifikasi skrip startup Anda untuk menyelesaikan dalam batas waktu yang dikonfigurasi.

RUN_ERROR

Anda mungkin mendapatkan ini ketika terjadi kesalahan saat menjalankan skrip komponen. Kode kesalahan dilaporkan dalam log komponen. Periksa skrip run untuk kesalahan dan deploy komponen Anda lagi.

RUN_MISSING_DEFAULT_RUNDENGAN

Anda mungkin mendapatkan kesalahan ini saat tidak AWS IoT Greengrass dapat menentukan pengguna atau grup yang akan digunakan saat menjalankan komponen. Periksa untuk memastikan bahwa `runWith` bagian resep run Anda menyertakan pengguna atau grup yang valid.

RUN_CONFIG_NOT_VALID

Anda mungkin mendapatkan kesalahan ini ketika komponen tidak dapat dijalankan karena `Run` bagian resep tidak valid. Periksa bagian run resep Anda untuk kesalahan dan coba penyebaran lagi.

RUN_IO_ERROR

Anda mungkin mendapatkan ini ketika kesalahan I/O terjadi saat komponen sedang berjalan. Periksa komponen kesalahan log untuk detail tentang kesalahan.

RUN_TIMEOUT

Anda mungkin mendapatkan kesalahan ini ketika skrip yang dijalankan tidak selesai dalam periode batas waktu yang dikonfigurasi. Baik meningkatkan `Timeout` periode yang ditentukan di `Run` bagian resep atau memodifikasi skrip run Anda untuk menyelesaikan dalam batas waktu yang dikonfigurasi.

SHUTDOWN_ERROR

Anda mungkin mendapatkan ini ketika terjadi kesalahan saat mematikan skrip komponen. Kode kesalahan dilaporkan dalam log komponen. Periksa skrip shutdown untuk kesalahan dan deploy komponen Anda lagi.

SHUTDOWN_TIMEOUT

Anda mungkin mendapatkan kesalahan ini ketika skrip shutdown tidak selesai dalam periode batas waktu yang dikonfigurasi. Baik meningkatkan `Timeout` periode yang ditentukan di `Shutdown` bagian resep atau memodifikasi skrip run Anda untuk menyelesaikan dalam batas waktu yang dikonfigurasi.

Beri tag pada sumber daya AWS IoT Greengrass Version 2 Anda

Dengan tanda, Anda dapat mengatur dan mengelola sumber daya Anda di AWS IoT Greengrass. Anda dapat menggunakan tag untuk menetapkan metadata ke sumber daya Anda, dan Anda dapat menggunakan tag dalam kebijakan IAM untuk menentukan akses bersyarat ke sumber daya Anda.

> **Note**
>
> Sekarang, Tanda sumber daya Greengrass tidak didukung untuk grup penagihan AWS IoT atau laporan alokasi biaya.

Menggunakan tag di AWS IoT Greengrass V2

Anda dapat menggunakan tag untuk mengategorikan sumber daya AWS IoT Greengrass Anda menurut tujuan, pemilik, lingkungan, atau klasifikasi lainnya untuk kasus penggunaan Anda. Bila Anda memiliki banyak sumber daya dengan jenis yang sama, tag akan membantu Anda dengan lebih mudah mengidentifikasi sumber daya tertentu.

Setiap tag terdiri atas sebuah kunci dan sebuah nilai opsional, yang keduanya Anda tentukan. Misalnya, Anda dapat menentukan serangkaian tanda untuk perangkat inti Anda yang dapat membantu Anda melacaknya oleh pelanggan yang memiliki perangkat tersebut. Kami menyarankan agar Anda merancang serangkaian kunci tanda yang memenuhi kebutuhan Anda untuk setiap jenis sumber daya. Dengan menggunakan serangkaian kunci tag, Anda dapat dengan lebih mudah mengelola sumber daya Anda.

Beri tag dengan AWS Management Console

Editor tanda di AWS Management Console menyediakan cara terpusat dan terpadu bagi Anda untuk membuat dan mengelola tanda Anda untuk sumber daya dari semua layanan AWS. Untuk informasi selengkapnya, lihat Editor tanda di Panduan Pengguna AWS Resource Groups.

Tanda dengan API AWS IoT Greengrass V2

Anda juga dapat menggunakan API AWS IoT Greengrass V2 untuk bekerja dengan tag. Sebelum Anda membuat tag, perhatikan pembatasan penandaan. Untuk informasi selengkapnya, lihat Penamaan dan konvensi penggunaan tanda di Referensi Umum AWS.

- Untuk menambahkan tanda selama pembuatan sumber daya, tentukan tag dalam properti `tags` sumber daya.
- Untuk menambahkan tanda ke sumber daya yang sudah ada, atau untuk memperbarui nilai tag, gunakan TagResourceoperasi.
- Untuk menghapus tag dari sebuah sumber daya, gunakan UntagResourceoperasi.
- Untuk mengambil tanda yang terkait dengan suatu sumber daya, gunakan ListTagsForResourceoperasi, atau jelaskan sumber daya tersebut dan periksa`tags` properti ini.

Tabel berikut mencantumkan sumber daya yang dapat Anda beri tag dengan menggunakan API AWS IoT Greengrass V2 dan operasi `Create` dan `Describe` atau `Get` yang sesuai.

Sumber daya AWS IoT Greengrass V2 yang dapat diberi tag

Resource	Buat operasi	Jelaskan atau dapatkan operasi
Perangkat inti	Tidak ada. Jalankan perangkat lunak inti AWS IoT Greengrass pada suatu perangkat untuk membuat perangkat inti.	GetCoreDevice
Komponen	CreateComponentVersion	DescribeComponent, GetComponent
Deployment	CreateDeployment	GetDeployment

Gunakan operasi berikut untuk melihat dan mengelola tag untuk sumber daya yang mendukung penandaan:

- TagResource— Menambahkan tag ke sumber daya, atau memperbarui nilai tag yang ada.
- ListTagsForResource- Daftar tag untuk sumber daya.

- UntagResource— Menghapus tag dari sebuah sumber daya.

Anda dapat menambahkan atau menghapus tanda pada sumber daya kapan saja. Untuk mengubah nilai kunci tanda, tambahkan tanda ke sumber daya yang menentukan kunci yang sama dan nilai yang baru. Nilai baru akan menggantikan 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, tanda-tanda yang berkaitan dengan sumber daya tersebut juga dihapus.

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.

> **Note**
>
> Jika Anda menggunakan tanda untuk mengizinkan atau menolak akses pengguna ke sumber daya, Anda harus menolak memberi pengguna kemampuan untuk menambahkan atau menghapus tanda tersebut untuk sumber daya yang sama. Jika tidak, pengguna dapat mengakali pembatasan Anda dan mendapatkan akses ke sumber daya dengan memodifikasi tandanya.

Anda dapat menggunakan kunci konteks syarat dan nilai-nilai berikut dalam `Condition`, yang juga disebut blok `Condition`, dari suatu pernyataan kebijakan.

`greengrassv2:ResourceTag/`*`tag-key`*`: `*`tag-value`*

Izinkan atau tolak tindakan pada sumber daya dengan tanda tertentu.

`aws:RequestTag/`*`tag-key`*`: `*`tag-value`*

Syaratkan tanda tertentu untuk digunakan, atau tidak digunakan, saat membuat atau mengubah sumber daya yang dapat ditandai.

`aws:TagKeys: [tag-key, ...]`

Syaratkan serangkaian kunci tanda tertentu untuk digunakan, atau tidak digunakan, saat membuat atau mengubah sumber daya yang dapat ditandai.

> **Note**
>
> Kunci dan nilai konteks syarat dalam kebijakan IAM hanya berlaku pada tindakan yang memiliki sumber daya yang dapat ditandai sebagai parameter wajib. Misalnya, Anda dapat mengatur akses bersyarat berbasis tag untuk ListCoreDevices.

Untuk informasi selengkapnya, lihat Mengontrol akses menggunakan tanda AWS dan Referensi kebijakan IAM JSON dalam Panduan Pengguna IAM.

Membuat sumber daya AWS IoT Greengrass dengan AWS CloudFormation

AWS IoT Greengrass terintegrasi dengan AWS CloudFormation, yaitu layanan yang membantu Anda membuat model dan mengatur sumber daya AWS agar Anda dapat menghemat waktu untuk membuat dan mengelola sumber daya dan infrastruktur Anda. Anda membuat template yang menjelaskan semuaAWSsumber daya yang Anda inginkan (seperti versi komponen dan penerapan), danAWS CloudFormationketentuan dan mengkonfigurasi sumber daya tersebut untuk Anda.

Saat menggunakan AWS CloudFormation, Anda dapat menggunakan kembali templat Anda untuk menyiapkan sumber daya AWS IoT Greengrass secara konsisten dan berulang kali. Jelaskan sumber daya Anda satu kali, lalu sediakan sumber daya yang sama berulang kali dalam beberapa Akun AWS dan Wilayah.

Templat AWS IoT Greengrass dan AWS CloudFormation

Untuk menyediakan dan mengonfigurasi sumber daya untuk AWS IoT Greengrass dan layanan terkait, Anda harus memahami templat AWS CloudFormation. Templat adalah file teks dengan format JSON atau YAML. Templat ini menjelaskan sumber daya yang ingin Anda sediakan di tumpukan AWS CloudFormation Anda. Jika Anda tidak terbiasa dengan JSON atau YAML, Anda dapat menggunakan AWS CloudFormation Designer untuk membantu Anda memulai dengan templat AWS CloudFormation. Untuk informasi selengkapnya, lihat Apa yang dimaksud dengan AWS CloudFormation Designer? dalam Panduan Pengguna AWS CloudFormation.

AWS IoT Greengrassmendukung pembuatan versi komponen dan penyebaran diAWS CloudFormation. Untuk informasi selengkapnya, termasuk contoh templat JSON dan YAKL untuk versi komponen dan deployment, lihatAWS IoT Greengrassreferensi tipe sumber dayadi dalamAWS CloudFormationPanduan Pengguna.

ComponentVersion Contoh templat

Berikut ini adalah template YAKL untuk versi komponen sederhana. Resep JSON mencakup jeda baris untuk dibaca.

```
Parameters:
  ComponentVersion:
    Type: String
```

```
Resources:
  TestSimpleComponentVersion:
    Type: AWS::GreengrassV2::ComponentVersion
    Properties:
      InlineRecipe: !Sub
        - "{\n
            \"RecipeFormatVersion\": \"2020-01-25\",\n
            \"ComponentName\": \"component1\",\n
            \"ComponentVersion\": \"${ComponentVersion}\",\n
            \"ComponentType\": \"aws.greengrass.generic\",\n
            \"ComponentDescription\": \"This\",\n
            \"ComponentPublisher\": \"You\",\n
            \"Manifests\": [\n
            {\n
              \"Platform\": {\n
              \"os\": \"darwin\"\n
            },\n
            \"Lifecycle\": {},\n
            \"Artifacts\": []\n
           },\n
           {\n
             \"Lifecycle\": {},\n
             \"Artifacts\": []\n
           }\n
          ],\n
          \"Lifecycle\": {\n
          \"install\": {\n
            \"script\": \"yuminstallpython\"\n
          }\n
         }\n
        }"
        - { ComponentVersion: !Ref ComponentVersion }
```

Contoh templat deployment

Berikut ini adalah file YAKL mendefinisikan template sederhana untuk penyebaran.

```
Parameters:
  ComponentVersion:
    Type: String
  TargetArn:
    Type: String
Resources:
```

```
  TestDeployment:
    Type: AWS::GreengrassV2::Deployment
    Properties:
      Components:
        component1:
          ComponentVersion: !Ref ComponentVersion
      TargetArn: !Ref TargetArn
      DeploymentName: CloudFormationIntegrationTest
      DeploymentPolicies:
        FailureHandlingPolicy: DO_NOTHING
        ComponentUpdatePolicy:
          TimeoutInSeconds: 5000
          Action: SKIP_NOTIFY_COMPONENTS
        ConfigurationValidationPolicy:
          TimeoutInSeconds: 30000
Outputs:
  TestDeploymentArn:
    Value: !Sub
      - arn:${AWS::Partition}:greengrass:${AWS::Region}:${AWS::AccountId}:deployments:
${DeploymentId}
      - DeploymentId: !GetAtt TestDeployment.DeploymentId
```

Pelajari selengkapnya tentang AWS CloudFormation

Untuk mempelajari selengkapnya tentang AWS CloudFormation, lihat sumber daya berikut:

- AWS CloudFormation
- AWS CloudFormationPanduan Pengguna
- AWS CloudFormationReferensi API
- AWS CloudFormationPanduan Pengguna Baris Perintah

Perangkat lunak inti AWS IoT Greengrass sumber terbuka

Waktu aktif edge AWS IoT Greengrass Version 2 (nukleus) dan komponen lain dari perangkat lunak inti AWS IoT Greengrass adalah sumber terbuka. Ini berarti bahwa Anda dapat meninjau kode untuk memecahkan masalah interaksi dengan aplikasi Anda. Anda juga dapat menyesuaikan dan memperpanjang perangkat lunak inti AWS IoT Greengrass untuk memenuhi kebutuhan perangkat lunak dan perangkat keras khusus Anda.

Untuk informasi tentang repositori open source untuk perangkat lunak AWS IoT Greengrass Core, lihat organisasi aws-greengrass di. GitHub Penggunaan perangkat lunak open source Anda diatur oleh lisensi open source di GitHub repositori yang sesuai.

Penggunaan Anda atas perangkat lunak inti AWS IoT Greengrass dan komponen yang tidak tunduk pada lisensi sumber terbuka diatur oleh Lisensi Perangkat Lunak AWS Greengrass Core.

Riwayat dokumen untuk Panduan AWS IoT Greengrass V2 Pengembang

Tabel berikut menjelaskan dokumentasi untuk rilis ini AWS IoT Greengrass Version 2.

- Versi API: 2020-11-30

Perubahan	Deskripsi	Tanggal
Manajer bayangan v2.3.8 dirilis	Shadow manager v2.3.8 tersedia. Rilis ini memperbai ki masalah di mana shadow manager menciptakan situasi kebuntuan selama koneksi klien MQTT.	Juni 5, 2024
Greengrass CLI v2.12.6 dirilis	Komponen CLI Greengrass v2.12.6 tersedia.	24 Mei 2024
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.12.6	Rilis ini menyediakan versi 2.12.6 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	24 Mei 2024
AWS IoT Device Tester v4.9.4 dengan GGV2Q v2.5.4 dirilis	Versi 4.9.4 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.5.4, dan mendukung Greengrass nucleus versi 2.12.0, 2.11.0, 2.10.0, 2.9.5.	3 Mei 2024
Terowongan aman v1.0.19 dirilis	Terowongan aman v1.0.19 tersedia. Versi ini memutakhi	1 Mei 2024

	rkan AWS IoT Device Client yang mendasari yang dipanggil oleh komponen dari versi 1.8.0 ke versi 1.9.0. Terowongan aman v1.0.19 meningkatkan batas terowongan bersamaan menjadi 20 terowongan pada tingkat komponen. Versi baru ini juga meningkatkan batas waktu AWS IoT Greengrass Core IPC dari 3 detik menjadi 10 detik.	
Konektor tepi untuk komponen Kinesis Video Streams v1.0.5 dirilis	Versi 1.0.5 dari konektor tepi untuk komponen Kinesis Video Streams tersedia. Versi ini mencakup perbaikan bug umum dan perbaikan.	April 29, 2024
Greengrass CLI v2.12.5 dirilis	Komponen CLI Greengrass v2.12.5 tersedia.	April 25, 2024
Komponen autentikasi perangkat klien v2.5.0 dirilis	Komponen autentikasi perangkat klien v2.5.0 tersedia. Rilis ini menambahk an dukungan variabel kebijakan untuk nama benda. Rilis ini juga memungkin kan sumber daya kebijakan dengan wildcard.	April 25, 2024
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.12.5	Rilis ini menyediakan versi 2.12.5 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	April 25, 2024

AWS IoT Device Tester v4.9.3 dengan GGV2Q v2.5.3 dirilis	Versi 4.9.3 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.5.3, dan mendukung Greengrass nucleus versi 2.12.0, 2.11.0, 2.10.0, 2.9.5.	April 5, 2024
Greengrass CLI v2.12.4 dirilis	Komponen CLI Greengrass v2.12.4 tersedia.	April 2, 2024
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.12.4	Rilis ini menyediakan versi 2.12.4 komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	April 2, 2024
Manajer bayangan v2.3.7 dirilis	Shadow manager v2.3.7 tersedia. Rilis ini memperbaiki masalah di mana manajer bayangan mencatat `NullPointerException` kesalahan secara berkala selama sinkronisasi manajer bayangan.	Maret 27, 2024
Moquette MQTT 3.1.1 broker v2.3.6 dirilis	Moquette MQTT 3.1.1 komponen broker v2.3.6 tersedia. Versi ini mencakup perbaikan bug umum dan perbaikan.	Maret 27, 2024
Konsol debug lokal v2.4.2 dirilis	Komponen konsol debug lokal v2.4.2 tersedia. Versi ini mencakup perbaikan bug umum dan perbaikan.	Maret 27, 2024

Manajer Lambda v2.3.3 dirilis	Komponen manajer Lambda v2.3.3 tersedia. Versi ini mencakup perbaikan bug umum dan perbaikan.	Maret 27, 2024
Detektor IP v2.1.9 dirilis	Komponen detektor IP v2.1.9 tersedia. Rilis ini menyesuai kan langkah yang diperoleh IP untuk hanya mengirim log pada tingkat log debug.	Maret 27, 2024
AWS IoT plugin penyediaan armada v1.2.1 dirilis	AWS IoT plugin penyediaa n armada v1.2.1 tersedia. Rilis ini memperbaiki masalah saat plugin penyediaan armada sedang offline selama startup inti Greengrass. Plugin penyediaan armada sekarang tanpa batas waktu mencoba ulang panggilan koneksi MQTT.	Maret 27, 2024
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.12.3	Rilis ini menyediakan versi 2.12.3 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	Maret 27, 2024
Greengrass CLI v2.12.3 dirilis	Komponen CLI Greengrass v2.12.3 tersedia.	Maret 25, 2024

AWS IoT Device Tester v4.9.2 dengan GGV2Q v2.5.2 dirilis	Versi 4.9.2 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.5.2, dan mendukung Greengrass nucleus versi 2.12.0, 2.11.0, 2.10.0, 2.9.5.	Maret 18, 2024
Lookout for Vision edge agent v1.2.0 dirilis	Lookout for Vision edge agent v1.2.0 tersedia.	Maret 11, 2024
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.12.2	Rilis ini menyediakan versi 2.12.2 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	Februari 15, 2024
Manajer bayangan v2.3.6 dirilis	Shadow manager v2.3.6 tersedia. Rilis ini memperbai ki masalah di mana properti bayangan yang dihapus melalui AWS Cloud pembaruan saat perangkat offline terus ada di bayangan lokal setelah mendapatkan kembali konektivitas.	Februari 14, 2024
Peluncur Lambda v2.0.13 dirilis	Versi 2.0.13 dari komponen peluncur Lambda tersedia. Rilis ini mencakup perbaikan bug umum dan perbaikan.	Februari 14, 2024

Disk spooler v1.0.3 dirilis	Komponen spooler disk v1.0.3 tersedia. Rilis ini meningkatkan kinerja dengan menggunakan kembali koneksi database.	Februari 14, 2024
Lookout for Vision edge agent v1.1.9 dirilis	Lookout for Vision edge agent v1.1.9 tersedia.	Januari 17, 2024
Kit Pengembangan Greengrass CLI v1.6.2	Versi 1.6.2 dari CLI Kit Pengembangan Greengrass tersedia. Versi ini memperbaiki masalah di mana Windows gradlew.bat tidak berfungsi karena jalur relatif. Versi ini juga berisi perbaikan tambahan.	Januari 16, 2024
Peristiwa CloudTrail data baru	Anda sekarang dapat mencatat peristiwa AWS CloudTrail data untuk mendapatkan informasi tentang operasi sumber daya seperti mendapatkan komponen atau konfigurasi penerapan. Gunakan acara ini untuk mendapatkan wawasan tentang pengoperasian perangkat Greengrass Anda.	20 Desember 2023
Lookout for Vision edge agent v1.1.8 dirilis	Lookout for Vision edge agent v1.1.8 tersedia.	Desember 12, 2023

Manajer aliran v2.1.12 dirilis	Stream manager v2.1.12 sekarang tersedia. Rilis ini mengubah urutan yang digunakan Greengrass untuk memilih satu set kredensional untuk panggilan layanan. AWS	8 Desember 2023
Jembatan MQTT v2.3.1 dirilis	Jembatan MQTT v2.3.1 tersedia. Rilis ini memperbaiki masalah langka di mana klien MQTT lokal masuk ke loop pemutusan.	8 Desember 2023
Disk spooler v1.0.2 dirilis	Komponen spooler disk v1.0.2 tersedia. Rilis ini memperbaiki masalah di mana bidang format pesan MQTT tidak bertahan dalam kasus tertentu.	8 Desember 2023
Komponen autentikasi perangkat klien v2.4.5 dirilis	Komponen autentikasi perangkat klien v2.4.5 tersedia. Rilis ini menambahk an dukungan untuk wildcard di akhir nama hal dalam aturan pemilihan dan memperbaiki masalah di mana sertifikat tidak diperbarui dengan info konektivitas baru dalam kasus tertentu.	8 Desember 2023
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.12.1	Rilis ini menyediakan versi 2.12.1 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	8 Desember 2023

Kit Pengembangan Greengrass CLI v1.6.1	Versi 1.6.1 dari CLI Kit Pengembangan Greengrass tersedia. Versi ini berisi perbaikan bug dan perbaikan.	6 Desember 2023
Validasi resep	Menambahkan fitur validasi resep yang akan memvalidasi resep komponen saat membuat versi komponen.	16 November 2023
Komponen yang didukung penerbit	AWS IoT Greengrass sekarang menawarkan komponen yang didukung Penerbit. Komponen-komponen ini dikembangkan, ditawarkan, dan dilayani oleh vendor pihak ketiga.	16 November 2023
Greengrass Testing Framework v1.2.0 dirilis	Greengrass Testing Framework v1.2.0 tersedia.	15 November 2023

Kit Pengembangan Greengrass CLI v1.6.0	Versi 1.6.0 dari CLI Kit Pengembangan Greengrass tersedia. Versi ini menambahkan pemeriksaan validasi resep terhadap skema resep Greengrass selama perintah and. `component build` `component publish` Pembaruan ini membantu pengembang mengidentifikasi masalah yang dapat ditindaklanjuti dalam resep komponen mereka sebelumnya dalam proses pembuatan komponen. Versi ini juga menambahkan suite uji kepercayaan ke template yang dapat ditarik ke bawah oleh `test-e2e init` perintah. Rangkaian uji kepercayaan diri ini mencakup delapan tes generik yang dapat digunakan dan diperluas agar sesuai dengan kebutuhan pengujian komponen dasar.	15 November 2023
AWS IoT Device Tester v4.9.1 mendukung Greengrass nucleus versi 2.12.0	Versi 4.9.1 dari IDT untuk AWS IoT Greengrass V2 sekarang mendukung Greengrass nucleus versi 2.12.0.	7 November 2023
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.12.0	Rilis ini menyediakan versi 2.12.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	7 November 2023

Mengoperasikan perangkat inti Greengrass di VPC	Mengoperasikan perangkat inti Greengrass di VPC tersedia. Fitur ini memungkinkan Anda untuk melakukan penyebaran di VPC tanpa akses internet publik.	3 November 2023
Greengrass CLI v2.12.0 dirilis	Komponen CLI Greengrass v2.12.0 tersedia.	30 Oktober 2023
Manajer aliran v2.1.10 dirilis	Stream manager v2.1.10 sekarang tersedia. Rilis ini memperbaiki masalah di mana konfigurasi proxy HTTPS tidak mempercayai rantai sertifikat Greengrass CA.	26 Oktober 2023
Peluncur Lambda v2.0.12 dirilis	Versi 2.0.12 dari komponen peluncur Lambda tersedia. Rilis ini memperbaiki masalah di mana peluncur Lambda dapat menimbulkan kesalahan jika proses sebelumnya tidak dihentikan dengan benar.	26 Oktober 2023

Kit Pengembangan Greengrass CLI v1.5.0	Versi 1.5.0 dari CLI Kit Pengembangan Greengrass tersedia. Versi ini memperbarui pola yang dikenali oleh opsi `excludes` build kapan `build_system adazip`. Versi ini sekarang akan mengenali pola glob yang cocok dengan nama jalur berdasarkan karakter wildcard mereka. Ini memungkinkan spesifikasi khusus direktori mana yang akan dikecualikan.	26 Oktober 2023
Lookout for Vision edge agent v1.1.7 dirilis	Lookout for Vision edge agent v1.1.7 tersedia.	24 Oktober 2023
Manajer bayangan v2.3.4 dirilis	Shadow manager v2.3.4 tersedia. Rilis ini menambahkan dukungan untuk dokumen status bayangan nol dan kosong.	18 Oktober 2023
Manajer log v2.3.6 dirilis	Komponen pengelola log v2.3.6 tersedia.	18 Oktober 2023
Konsol debug lokal v2.4.0 dirilis	Komponen konsol debug lokal v2.4.0 tersedia.	18 Oktober 2023
Manajer Lambda v2.3.1 dirilis	Komponen manajer Lambda v2.3.1 tersedia.	18 Oktober 2023
Greengrass CLI v2.11.3 dirilis	Komponen CLI Greengrass v2.11.3 tersedia.	18 Oktober 2023

AWS IoT Greengrass Pembaruan perangkat lunak Core v2.11.3	Rilis ini menyediakan versi 2.11.3 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	18 Oktober 2023
Terowongan aman v1.0.17 dirilis	Terowongan aman v1.0.17 tersedia.	4 Oktober 2023
Kit Pengembangan Greengrass CLI v1.4.0	Versi 1.4.0 dari Greengrass Development Kit CLI tersedia. Versi ini menambahkan `config` perintah baru yang memulai prompt interaktif untuk memodifikasi bidang dalam file konfigurasi GDK yang ada. Versi ini juga memodifikasi `gdk component publish` perintah `gdk component build` dan untuk memverifikasi bahwa ukuran resep berada dalam persyaratan Greengrass (<=16000 byte) sebelum melanjutkan.	2 Oktober 2023
Moquette MQTT 3.1.1 broker v2.3.5 dirilis	Moquette MQTT 3.1.1 komponen broker v2.3.5 tersedia. Versi ini memperbarui Moquette ke versi 0.17.	28 September 2023
Jembatan MQTT v2.3.0 dirilis	Jembatan MQTT v2.3.0 tersedia. Rilis ini menambahkan dukungan MQTT 5 untuk menjembatani antara AWS IoT Core dan sumber MQTT lokal.	28 September 2023

Lookout for Vision edge agent v1.1.6 dirilis	Lookout for Vision edge agent v1.1.6 tersedia.	27 September 2023
Manajer Lambda v2.3.0 dirilis	Komponen manajer Lambda v2.3.0 tersedia.	15 September 2023
Peluncur Lambda v2.0.11 dirilis	Versi 2.0.11 dari komponen peluncur Lambda tersedia. Versi ini mendukung Lambda Manager 2.3.0.	15 September 2023
Moquette MQTT 3.1.1 broker v2.3.4 dirilis	Moquette MQTT 3.1.1 komponen broker v2.3.4 tersedia.	1 September 2023
Kerangka Pengujian Greengrass	GTF adalah kumpulan blok bangunan untuk mendukung end-to-end otomatisasi. Ini memungkinkan pelanggan AWS IoT Greengrass Version 2 internal untuk menggunakan kerangka pengujian yang sama dengan yang digunakan tim layanan untuk perubahan perangkat lunak yang memenuhi syarat, penerimaan otomatis, dan tujuan jaminan kualitas.	11 Agustus 2023
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.11.2	Rilis ini menyediakan versi 2.11.2 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	9 Agustus 2023

Kit Pengembangan Greengrass CLI v1.3.0	Versi 1.3.0 dari Greengrass Development Kit CLI tersedia. Versi ini menambahkan `test-e2e` perintah baru untuk mendukung end-to-end pengujian komponen menggunakan Open Test Framework.	21 Juli 2023
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.11.1	Rilis ini menyediakan versi 2.11.1 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	21 Juli 2023
Disk spooler v1.0.0 dirilis	Komponen spooler disk v1.0.0 tersedia.	28 Juni 2023
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.11.0	Rilis ini menyediakan versi 2.11.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	28 Juni 2023
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.10.3	Rilis ini menyediakan versi 2.10.3 komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	Juni 21, 2023
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.10.2	Rilis ini menyediakan versi 2.10.2 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	Juni 5, 2023

AWS IoT Greengrass Pembaruan perangkat lunak Core v2.10.1	Rilis ini menyediakan versi 2.10.1 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	11 Mei 2023
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.10.0	Rilis ini menyediakan versi 2.10.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	9 Mei 2023
SageMaker Manajer Edge dihentikan	Komponen Amazon SageMaker Edge Manager dihentikan pada 26 April 2024.	28 April 2023
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.9.6	Rilis ini menyediakan versi 2.9.6 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	20 April 2023
Manajer log v2.3.2 dirilis	Komponen pengelola log v2.3.2 tersedia.	19 April 2023

Manajer aliran v2.1.4 dirilis	Stream manager v2.1.4 sekarang tersedia. Rilis ini memperbaiki masalah saat entri untuk aset properti yang sama dengan stempel waktu yang sama dalam satu batch kembali `ConflictingOperationException` dari SiteWise API yang menyebabkan pengelola aliran terus mencoba lagi. Rilis ini juga memperbarui batas waktu koneksi default dari 3 detik hingga 1 menit.	13 April 2023
Kit Pengembangan Greengrass CLI v1.2.3	Versi 1.2.3 dari CLI Kit Pengembangan Greengrass tersedia. Versi ini berisi perbaikan bug.	13 April 2023
Komponen autentikasi perangkat klien v2.4.0 dirilis	Komponen autentikasi perangkat klien v2.4.0 tersedia. Rilis ini menambahkan dukungan untuk autentikasi perangkat klien untuk memancarkan metrik operasional yang dapat ditampilkan di dasbor Perangkat Klien Greengrass.	April 10, 2023
Kit Pengembangan Greengrass CLI v1.2.2	Versi 1.2.2 dari CLI Kit Pengembangan Greengrass tersedia. Versi ini berisi perbaikan dan perbaikan bug.	April 7, 2023

AWS IoT Greengrass Pembaruan perangkat lunak Core v2.9.5	Rilis ini menyediakan versi 2.9.5 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	30 Maret 2023
Manajer aliran v2.1.3 dirilis	Stream manager v2.1.3 sekarang tersedia. Rilis ini memperbaiki masalah startup pada OS Windows saat berjalan sebagai pengguna SYSTEM.	7 Maret 2023
Adaptor protokol Modbus-RTU v2.1.5 dirilis	Komponen adaptor protokol Modbus-RTU v2.1.5 tersedia. Rilis ini memperbaiki masalah dengan `ReadDiscreteInput` operasi.	7 Maret 2023
Komponen autentikasi perangkat klien v2.3.2 dirilis	Komponen autentikasi perangkat klien v2.3.2 tersedia. Rilis ini menambahkan dukungan untuk caching informasi hostname sehingga komponen menghasilkan subjek sertifikat dengan benar saat di-restart saat offline.	7 Maret 2023
AWS IoT Device Tester v4.7.0 mendukung Greengrass nucleus versi 2.9.4	Versi 4.7.0 dari IDT untuk AWS IoT Greengrass V2 sekarang mendukung Greengrass nucleus versi 2.9.4.	2 Maret 2023
Antarmuka Baris Perintah Greengrass v1.2.0 dirilis	Greengrass Command Line Interface v1.2.0 tersedia.	28 Februari 2023

AWS IoT Greengrass Pembaruan perangkat lunak Core v2.9.4	Rilis ini menyediakan versi 2.9.4 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	Februari 24, 2023
Manajer bayangan v2.3.1 dirilis	Shadow manager v2.3.1 tersedia. Rilis ini memperbaiki kondisi yang dapat mencegah sinkronisasi pembaruan bayangan awan. Rilis ini juga memperbaiki masalah di mana perubahan konfigura si sinkronisasi bayangan bernama hanya berlaku untuk satu bayangan bernama.	21 Februari 2023
AWS IoT Device Tester v4.7.0 mendukung Greengrass nucleus versi 2.9.3	Versi 4.7.0 dari IDT untuk AWS IoT Greengrass V2 sekarang mendukung Greengrass nucleus versi 2.9.3.	9 Februari 2023
Praktik terbaik IAM diperbarui	Panduan yang diperbarui untuk menyelaraskan dengan praktik terbaik IAM. Untuk informasi selengkapnya, lihat Praktik terbaik keamanan di IAM.	3 Februari 2023
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.9.3	Rilis ini menyediakan versi 2.9.3 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	1 Februari 2023
Manajer log v2.3.1 dirilis	Manajer log v2.3.1 tersedia.	27 Januari 2023

AWS IoT Device Tester v4.7.0 mendukung Greengrass nucleus versi 2.9.2	Versi 4.7.0 dari IDT untuk AWS IoT Greengrass V2 sekarang mendukung Greengrass nucleus versi 2.9.2.	Januari 3, 2023
Manajer bayangan v2.3.0 dirilis	Shadow manager v2.3.0 tersedia. Rilis ini memperbai ki masalah yang mungkin mencegah sinkronisasi bayangan saat perangkat menyimpan kunci pribadi perangkat Greengrass dalam modul keamanan perangkat keras.	Desember 29, 2022
AWS IoT plugin penyediaan armada v1.2.0 dirilis	AWS IoT plugin penyediaan armada v1.2.0 tersedia. Rilis ini menambahkan dukungan untuk penyediaan perangkat melalui permintaan penandata nganan sertifikat dengan jalur kunci pribadi yang dapat dikonfigurasi.	22 Desember 2022
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.9.2	Rilis ini menyediakan versi 2.9.2 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	22 Desember 2022

AWS IoT Device Tester v4.7.0 dengan GGV2Q v2.5.0 dirilis	Versi 4.7.0 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.5.0, dan mendukung Greengrass nucleus versi 2.9.1, 2.9.0, 2.8.1, 2.8.0, 2.7.0, dan 2.6.0.	13 Desember 2022
Manajer bayangan v2.2.4 dirilis	Memperbaiki masalah ketika validasi ukuran bayangan tidak konsisten dengan cloud saat memperbarui dokumen bayangan lokal. Ini juga memperbaiki masalah di mana pengelola bayangan berhenti mendengarkan pembaruan konfigurasi jika penerapan melakukan a RESET pada node konfigurasi.	Desember 8, 2022
Lookout for Vision Edge Agent 1.1.1 dirilis	Komponen Lookout for Vision Edge Agent v1.1.1 tersedia.	Desember 5, 2022
Manajer log v2.3.0 dirilis	Komponen pengelola log v2.3.0 tersedia.	18 November 2022
AWS IoT Device Tester v4.5.11 mendukung Greengrass nucleus versi 2.9.1	Versi 4.5.11 dari IDT untuk AWS IoT Greengrass V2 sekarang mendukung Greengrass nucleus versi 2.9.1.	18 November 2022

AWS IoT Greengrass Pembaruan perangkat lunak Core v2.9.1	Rilis ini menyediakan versi 2.9.1 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	18 November 2022
AWS IoT Device Tester v4.5.11 mendukung Greengrass nucleus versi 2.9.0	Versi 4.5.11 dari IDT untuk AWS IoT Greengrass V2 sekarang mendukung Greengrass nucleus versi 2.9.0.	17 November 2022
Manajer aliran v2.1.2 dirilis	Stream manager v2.1.2 sekarang tersedia. Rilis ini memperbaiki masalah pada OS Windows yang menggunakan bahasa non-Inggris.	15 November 2022
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.9.0	Rilis ini menyediakan versi 2.9.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	15 November 2022
AWS IoT Device Tester v4.5.11 mendukung Greengrass nucleus versi 2.8.1	Versi 4.5.11 dari IDT untuk AWS IoT Greengrass V2 sekarang mendukung Greengrass nucleus versi 2.8.1.	Oktober 19, 2022

AWS IoT Device Tester v4.5.11 dengan GGV2Q v2.4.1 dirilis	Versi 4.5.11 IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.4.1, dan mendukung Greengrass nucleus versi 2.8.0, 2.7.0, dan 2.6.0.	13 Oktober 2022
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.8.1	Rilis ini menyediakan versi 2.8.1 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	13 Oktober 2022
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.8.0	Rilis ini menyediakan versi 2.8.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS	Oktober 7, 2022
Menambahkan AWS CloudFormation dukungan untuk penerapan	AWS CloudFormation sekarang mendukung AWS IoT Greengrass penerapan sebagai sumber daya.	6 Oktober 2022

SageMaker Edge Manager v1.3.0 dirilis	Komponen Amazon SageMaker Edge Manager v1.3.0 tersedia. Rilis ini menambahkan dukungan untuk komponen ini untuk mengatur ukuran disk untuk cache model TensorRT, dan meningkatkan konkurensi prediksi untuk memanfaatkan mesin akselerator perangkat seperti GPU dengan lebih baik.	September 1, 2022
Gunakan komunikasi antarproses (IPC) klien V2	Menambahkan informasi tentang klien IPC V2, yang mengurangi jumlah kode yang perlu Anda tulis untuk menggunakan operasi IPC dan membantu menghindari kesalahan umum yang dapat terjadi dengan klien IPC V1.	12 Agustus 2022
AWS IoT Device Tester v4.5.8 dengan GGV2Q v2.4.0 dirilis	Versi 4.5.8 IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.4.0, dan mendukung Greengrass nucleus versi 2.7.0, 2.6.0, dan 2.5.6.	12 Agustus 2022

SageMaker Edge Manager v1.2.0 dirilis	Komponen Amazon SageMaker Edge Manager v1.2.0 tersedia. Rilis ini menambahkan dukungan untuk komponen ini untuk secara otomatis mengambil model yang SageMaker dikompilasi NEO yang Anda unggah ke Amazon S3, sehingga Anda dapat menerapkan model baru tanpa perlu membuat penerapan. AWS IoT Greengrass	3 Agustus 2022
AWS IoT Device Tester v4.5.3 mendukung Greengrass nucleus versi 2.7.0	Versi 4.5.3 dari IDT untuk AWS IoT Greengrass V2 sekarang mendukung Greengrass nucleus versi 2.7.0.	1 Agustus 2022
Manajer aliran v2.1.0 dirilis	Stream manager v2.1.0 sekarang tersedia. Rilis ini mencakup dukungan bagi Anda untuk mengirim metrik telemetri ke Amazon. EventBridge	28 Juli 2022
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.7.0	Rilis ini menyediakan versi 2.7.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Ini termasuk dukungan bagi Anda untuk mengirim metrik telemetri ke Amazon. EventBridge	28 Juli 2022

SiteWise Penerbit IoT v2.2.0 dirilis	Komponen SiteWise penerbit IoT v2.2.0 tersedia. Rilis ini memperbarui komponen untuk memampatkan data sebelum mengirimkannya ke AWS IoT SiteWise layanan, yang mengurangi penggunaan bandwidth hingga 75 persen.	19 Juli 2022
Tutorial: Kembangkan komponen yang berinteraksi dengan bayangan perangkat klien	Menambahkan modul baru ke Tutorial: Berinteraksi dengan perangkat IoT lokal melalui MQTT yang dapat Anda ikuti untuk mempelajari cara mengembangkan komponen yang berinteraksi dengan bayangan perangkat klien.	18 Juli 2022
Pilih broker MQTT lokal	Menambahkan informasi tentang cara memilih broker MQTT lokal tempat perangkat klien terhubung ke perangkat inti.	18 Juli 2022
AWS IoT Device Tester v4.5.3 mendukung Greengrass nucleus versi 2.6.0	Versi 4.5.3 dari IDT untuk AWS IoT Greengrass V2 sekarang mendukung Greengrass nucleus versi 2.6.0.	Juni 29, 2022

AWS IoT Greengrass Pembaruan perangkat lunak Core v2.6.0	Rilis ini menyediakan versi 2.6.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Ini termasuk dukungan untuk bayangan perangkat klien dan broker MQTT 5 lokal untuk perangkat klien. Ini juga mencakup dukungan untuk wildcard dalam topik penerbitan/berlangganan lokal, variabel resep dalam konfigurasi komponen, dan wildcard dalam kebijakan otorisasi IPC. Fitur-fitur ini memungkinkan Anda untuk lebih mudah mengembangkan dan mengkonfigurasi komponen yang Anda terapkan ke armada perangkat inti. Rilis ini juga mencakup dukungan untuk komponen untuk menggunakan operasi IPC yang mengelola penyebaran lokal dan komponen pada perangkat inti.	Juni 27, 2022

Pembaruan komponen perangkat klien	Auth perangkat klien v2.1.0, broker MQTT (Moquette) v2.1.0, jembatan MQTT v2.1.1, dan detektor IP v2.1.2 tersedia. Rilis ini meningkatkan rotasi sertifikat, meningkatkan kinerja broker MQTT, dan memperbaiki masalah dengan cara komponen ini menangani pembaruan pengaturan ulang konfigurasi.	14 Juni 2022
AWS IoT Device Tester v4.5.3 mendukung Greengrass nucleus versi 2.5.6	Versi 4.5.3 dari IDT untuk AWS IoT Greengrass V2 sekarang mendukung Greengrass nucleus versi 2.5.6.	1 Juni 2022
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.5.6	Rilis ini menyediakan versi 2.5.6 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Ini termasuk dukungan untuk modul keamanan perangkat keras dengan kunci ECC. Ini juga mencakup perbaikan dan peningkatan bug lainnya.	31 Mei 2022
AWS IoT plugin penyediaan armada v1.1.0 dirilis	AWS IoT plugin penyediaan armada v1.1.0 tersedia. Rilis ini menambahkan dukungan untuk format jalur file tambahan ketika Anda mengkonfigurasi plugin pada perangkat Windows.	12 Mei 2022

Runtime Lambda baru dirilis	Menambahkan dukungan untuk runtime Lambda baru: Python 3.9, Java 11, dan NodeJS 14.	10 Mei 2022
Kembangkan komponen Greengrass yang menunda pembaruan komponen	Menambahkan tutorial yang dapat Anda ikuti untuk mempelajari cara mengembangkan komponen Greengrass yang menghalangi pembaruan komponen dari penerapan. Anda mungkin ingin menunda pembaruan ketika perangkat memiliki tingkat baterai rendah atau saat menjalankan proses yang tidak dapat terganggu, misalnya.	4 Mei, 2022
CloudWatch metrik v3.1.0 dan v3.1.0 dirilis AWS IoT Device Defender	CloudWatch komponen metrik v3.1.0 dan AWS IoT Device Defender komponen v3.1.0 tersedia. Rilis ini menambahkan dukungan untuk konfigurasi proxy jaringan HTTPS. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proxy jaringan dan Aktifkan perangkat inti untuk mempercayai proxy HTTPS.	27 April 2022
Migrasi dari AWS IoT Greengrass Version 1	Menambahkan panduan yang dapat Anda ikuti untuk bermigrasi dari AWS IoT Greengrass V1 ke AWS IoT Greengrass V2.	26 April 2022

AWS IoT Device Tester v4.5.3 dengan GGV2Q v2.3.1 diperbarui dan IDT v4.5.1 dengan GGV2Q v2.3.0 ditambahkan ke versi yang didukung	Versi 4.5.3 dari IDT untuk AWS IoT Greengrass V2 dengan AWS IoT Greengrass suite kualifikasi V2 (GGV2Q) v2.3.1 telah diperbarui untuk menyertakan dukungan untuk Greengrass nucleus versi 2.5.5, 2.5.4, dan 2.5.3. Pembaruan ini juga mencakup IDT 4.5.1 dengan suite kualifikasi AWS IoT Greengras s V2 (GGV2Q) v2.3.0 sebagai versi yang didukung. IDT 4.5.1 dengan suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.3.0 mendukung Greengrass nucleus versi 2.5.3.	April 25, 2022
Adaptor protokol Modbus-RTU v2.1.0 dirilis	Komponen adaptor protokol Modbus-RTU v2.1.0 tersedia. Rilis ini menambahkan parameter baru yang dapat Anda tentukan untuk mengkonfigurasi komunikas i serial dengan perangkat Modbus RTU.	20 April 2022

CloudWatch metrik v2.1.0, Firehose v2.1.0, dan Amazon SNS v2.1.0 dirilis	CloudWatch komponen metrik v2.1.0, komponen Firehose v2.1.0, dan komponen Amazon SNS v2.1.0 tersedia. Rilis ini menambahkan dukungan untuk konfigurasi proxy jaringan HTTPS. Untuk informasi selengkapnya, lihat Connect pada port 443 atau melalui proxy jaringan dan Aktifkan perangkat inti untuk mempercayai proxy HTTPS.	19 April 2022
AWS IoT Device Tester v4.5.3 dengan GGV2Q v2.3.1 dirilis	Versi 4.5.3 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.3.1, dan mendukung Greengrass nucleus versi 2.5.5.	15 April 2022

AWS IoT Greengrass Pembaruan perangkat lunak Core v2.5.5	Rilis ini menyediakan versi 2.5.5 dari komponen inti Greengrass dan komponen yang disediakan pembaruan . AWS Ini menambahkan dukungan untuk perangkat Windows yang menggunak an bahasa tampilan selain bahasa Inggris. Ini juga memperbaiki masalah di mana perangkat inti tidak melaporka n statusnya ke layanan AWS IoT Greengrass cloud setelah penyediaan dalam skenario tertentu.	April 6, 2022
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.5.4	Rilis ini menyediakan versi 2.5.4 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Ini termasuk perbaikan bug dan peningkatan.	Maret 23, 2022
Unduh secara AWS IoT Device Tester terprogram	Menambahkan informasi tentang cara mengunduh IDT secara AWS IoT Greengrass V2 terprogram.	15 Maret 2022

Kit Pengembangan Greengrass CLI v1.1.0	Versi 1.1.0 dari Greengrass Development Kit CLI tersedia. Versi ini menambahkan argumen baru ke `component publish` perintah `component init` dan. Versi ini juga memperbarui `component publish` perintah untuk membangun komponen jika tidak dibangun.	Februari 24, 2022
Manajer bayangan v2.1.0 dirilis	Komponen manajer bayangan v2.1.0 tersedia. Rilis ini menambahkan opsi untuk mengonfigurasi interval di mana komponen menyinkronkan bayangan dengan AWS IoT Core. Misalnya, Anda dapat menentukan interval yang lebih lama untuk mengurangi penggunaan dan biaya bandwidth.	3 Februari, 2022
Gambar Dockerfile dan Docker untuk AWS IoT Greengrass perangkat lunak Core v2.5.3	Gambar Dockerfile dan Docker untuk perangkat lunak AWS IoT Greengrass Core v2.5.3 sekarang tersedia.	12 Januari 2022

AWS IoT Device Tester v4.5.1 dengan GGV2Q v2.3.0 dirilis	Versi 4.5.1 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.3.0, dan mendukung validasi dan kualifikasi perangkat berbasis Linux yang menggunakan modul keamanan perangkat keras (HSM) untuk menyimpan kunci pribadi dan sertifika t yang digunakan oleh perangkat lunak Core. AWS IoT Greengrass	11 Januari 2022
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.5.3	Rilis ini menyediakan versi 2.5.3 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Ini termasuk dukungan bagi Anda untuk mengkonfi gurasi perangkat lunak AWS IoT Greengrass Core untuk menggunakan kunci pribadi dan sertifikat yang Anda simpan dengan aman dalam modul keamanan perangkat keras (HSM).	6 Januari 2022
Gambar Dockerfile dan Docker untuk AWS IoT Greengrass perangkat lunak Core v2.5.2	Gambar Dockerfile dan Docker untuk perangkat lunak AWS IoT Greengrass Core v2.5.2 sekarang tersedia.	Desember 20, 2021

AWS IoT Device Tester v4.4.1 dengan GGV2Q v2.2.1 dirilis	Versi 4.4.1 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.2.1, dan mendukung Greengrass nucleus versi 2.5.2 untuk kualifikasi perangkat.	Desember 12, 2021
Lakukan inferensi pembelajaran mesin menggunakan Amazon Lookout for Vision	Menambahkan informasi tentang cara melakukan inferensi pembelajaran mesin menggunakan Lookout for Vision pada perangkat inti Greengrass. Lookout for Vision menggunakan visi komputer untuk menemukan cacat visual pada produk industri.	8 Desember 2021
AWS IoT Device Tester v4.4.1 dengan GGV2Q v2.2.0 dirilis	Versi 4.4.1 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.2.0, dan mendukung Greengrass nucleus versi 2.5.2 untuk kualifikasi perangkat.	Desember 6, 2021

AWS IoT Greengrass Pembaruan perangkat lunak Core v2.5.2	Rilis ini menyediakan versi 2.5.2 dari komponen inti Greengrass dan komponen yang disediakan pembaruan . AWS Ini memperbaiki masalah dengan layanan Windows yang terjadi setelah pembaruan inti Greengrass. Ini juga termasuk dukungan untuk AWS IoT Device Defender komponen pada perangkat Windows.	Desember 3, 2021
Konektor tepi baru untuk komponen Kinesis Video Streams	Versi 1.0.0 dari konektor tepi untuk komponen Kinesis Video Streams tersedia. Yang AWS disediakan ini membaca umpan video dari kamera lokal dan menerbitkan aliran ke Kinesis Video Streams. Komponen ini terintegrasi dengan AWS IoT TwinMaker , yang memungkinkan Anda untuk melihat dan mengelola aliran video dan data lainnya di dasbor Grafana.	30 November 2021

Kelola perangkat inti Greengrass dengan AWS Systems Manager	Menambahkan informasi tentang cara mengelola perangkat inti Greengras s dengan. AWS Systems Manager Systems Manager adalah AWS layanan yang memungkinkan Anda melihat data operasional, mengotoma tiskan tugas operasi, dan menjaga keamanan dan kepatuhan.	29 November 2021
Kit Pengembangan Greengras s CLI	Menambahkan informasi tentang AWS IoT Greengrass Development Kit Command-L ine Interface (GDK CLI), yang merupakan alat yang dapat Anda unduh ke komputer pengembangan lokal Anda untuk membantu Anda mengembangkan komponen Greengrass kustom. Anda dapat menggunakan CLI GDK untuk membuat, membangun , dan menerbitkan komponen kustom.	29 November 2021

Komponen Greengrass yang disediakan komunitas	Menambahkan informasi tentang Katalog Perangkat Lunak Greengrass, yang merupakan indeks komponen Greengrass yang dikembang kan oleh komunitas Greengras s. Dari katalog ini, Anda dapat mengunduh, memodifikasi, dan menyebarkan komponen untuk membuat aplikasi Greengrass Anda.	29 November 2021
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.5.1	Rilis ini menyediakan versi 2.5.1 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Ini termasuk dukungan untuk Java 32-bit pada perangkat Windows. Ini juga memperbaiki masalah dengan perilaku penghapusan grup hal baru dan memuat variabel lingkungan sistem pada perangkat Windows.	23 November 2021
AWS IoT Device Tester v4.4.0 dengan GGV2Q v2.1.0 dirilis	Versi 4.4.0 IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengras s V2 (GGV2Q) v2.1.0, dan mendukung kualifika si perangkat Greengras s berbasis Windows yang menjalankan Greengrass nucleus versi 2.5.0.	November 19, 2021

AWS IoT Greengrass Pembaruan perangkat lunak Core v2.5.0	Rilis ini menyediakan versi 2.5.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Ini termasuk dukungan untuk menjalankan perangkat lunak AWS IoT Greengras s Core pada perangkat Windows. Itu juga mengubah perilaku penghapusan grup benda dan menambahkan dukungan untuk proxy HTTPS.	12 November 2021
SageMaker Edge Manager v1.1.0 dirilis	Komponen Amazon SageMaker Edge Manager v1.1.0 tersedia. Rilis ini menambahkan dukungan untuk perangkat inti Greengras s yang menjalankan Amazon Linux 2, dan menambahkan parameter konfigurasi baru untuk menentukan lokasi folder data pengambilan di perangkat Anda.	3 November 2021
Pembaruan pencegahan wakil kebingungan lintas layanan	AWS IoT Greengrass V2 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

Pembaruan komponen perangkat klien	Perangkat klien auth v2.0.3, detektor IP v2.1.0, jembatan MQTT v2.1.0, dan broker MQTT (Moquette) v2.0.2 tersedia. Rilis ini menambahkan dukungan penuh untuk port broker MQTT non-default dan mencakup perbaikan dan peningkatan bug lainnya.	28 Oktober 2021
Manajer bayangan v2.0.4 dirilis	Komponen manajer bayangan v2.0.4 tersedia. Rilis ini memperbaiki masalah yang menyebabkan shadow manager menghapus versi yang baru dibuat dari bayangan apa pun yang sebelumnya dihapus. Dimulai dengan rilis ini, operasi `DeleteThingShadow` IPC menambah versi bayangan.	20 Oktober 2021
Manajer log v2.2.0 dirilis	Komponen pengelola log v2.2.0 tersedia. Manajer log sekarang mendukung penggunaan peta konfigurasi untuk menyediakan konfigurasi log komponen.	20 Oktober 2021
Manajer Lambda v2.1.4 dirilis	Komponen manajer Lambda v2.1.4 tersedia. Rilis ini memperbaiki masalah yang menyebabkan fungsi Lambda yang menggunakan runtime NodeJS hanya memproses satu pesan.	20 Oktober 2021

Gunakan komunikasi antarproses, AWS kredensi, dan manajer aliran di komponen kontainer Docker	Menambahkan informasi tentang cara menggunak an komunikasi antarproses (IPC), AWS kredensial, dan pengelola aliran di komponen kontainer Docker kustom Anda.	19 Oktober 2021
Komponen pemancar telemetri nukleus baru	Versi 1.0.0 dari komponen pemancar telemetri nukleus tersedia. Komponen AWS yang disediakan ini mengumpulkan data telemetri kesehatan sistem dan menerbitkannya terus menerus ke topik lokal dan topik MQTT. AWS IoT Core	30 September 2021
Izinkan lalu lintas perangkat melalui proxy atau firewall	Menambahkan informasi tentang titik akhir dan port yang digunakan perangkat inti Greengrass, sehingga Anda dapat membatasi lalu lintas sebagai langkah pengamanan.	September 16, 2021
AWS IoT Device Tester v4.2.0 dengan GGV2Q v2.0.1 dirilis	Versi 4.2.0 dari IDT untuk AWS IoT Greengrass V2 telah diperbarui dengan suite kualifikasi V2 (AWS IoT Greengrass GGV2Q) v2.0.1. Rilis ini mendukung Greengras s nucleus versi 2.4.0 untuk kualifikasi perangkat.	31 Agustus 2021

Komponen penginstal pembelajaran mesin yang diperbarui	Komponen penginstal DLR v1.6.5 dan komponen penginstal TensorFlow Lite v2.5.4 tersedia. Versi komponen ini menyertakan parameter `UseInstaller` konfigurasi baru yang memungkinkan Anda menonaktifkan skrip instalasi default.	Agustus 30, 2021
Dukungan Linux tertanam untuk AWS IoT Greengrass	BitBake Resep untuk AWS IoT Greengrass V2 tersedia dalam `meta-aws` proyek di GitHub. Anda dapat menggunakan resep ini untuk membangun sistem operasi berbasis Linux khusus menggunakan Yocto Project.	Agustus 20, 2021
Integritas kode	Menambahkan informasi tentang cara AWS IoT Greengrass V2 memverifikasi integritas perangkat lunak yang diunduh perangkat inti Greengrass dari. AWS Cloud	19 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

Manajer aliran v2.0.12 dirilis	Stream manager v2.0.12 sekarang tersedia. Rilis ini memperbaiki masalah yang mencegah peningkatan dari versi 2.0.7 komponen pengelola aliran ke versi antara v2.0.8 dan v2.0.11.	Agustus 10, 2021
Gambar Dockerfile dan Docker untuk AWS IoT Greengrass perangkat lunak Core v2.4.0	Gambar Dockerfile dan Docker untuk perangkat lunak AWS IoT Greengrass Core v2.4.0 sekarang tersedia.	9 Agustus 2021
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.4.0	Rilis ini menyediakan versi 2.4.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Ini termasuk dukungan untuk batas sumber daya sistem komponen, operasi IPC untuk menjeda dan melanjutk an komponen, dan menyediak an plugin.	Agustus 3, 2021
AWS IoT SiteWise Komponen baru	Menambahkan komponen yang AWS disediakan berikut untuk AWS IoT SiteWise: kolektor_ SiteWise IoT OPC-UA, penerbit IoT, dan SiteWise prosesor IoT. SiteWise	29 Juli 2021

AWS IoT Device Tester v4.2.0 dengan GGV2Q v2.0.0 dirilis	Versi 4.2.0 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v2.0.0 dan mencakup dukungan untuk tes kualifikasi opsional untuk komponen Docker, pembelajaran mesin, dan manajer aliran.	14 Juli 2021
AWS IoT Greengrass Pustaka IPC inti tersedia AWS IoT Device SDK untuk C++ v2	Versi 1.13.0 AWS IoT Device SDK untuk C++ v2 mendukung AWS IoT Greengrass Core IPC, sehingga Anda dapat mengembangkan komponen dalam C ++ yang berinteraksi dengan perangkat lunak Core. AWS IoT Greengrass	14 Juli 2021
SageMaker Komponen Edge Manager v1.0.2 dirilis	Komponen Amazon SageMaker Edge Manager v1.0.2 tersedia. Rilis ini memperbarui skrip instalasi dalam siklus hidup komponen. Perangkat inti Anda sekarang pasti memiliki Python 3.6 atau yang lebih baru, termasuk `pip` untuk versi Python Anda, yang diinstal pada perangkat sebelum Anda men-deploy komponen ini.	12 Juli 2021

Support update untuk AWS IoT Device Tester untuk AWS IoT Greengrass V2	IDT untuk AWS IoT Greengrass V2 versi 4.1.0 sekarang mendukung penggunaan Greengrass nucleus versi 2.3.0 untuk kualifikasi perangkat.	8 Juli 2021
Gambar Dockerfile dan Docker untuk AWS IoT Greengrass perangkat lunak Core v2.3.0	Gambar Dockerfile dan Docker untuk perangkat lunak AWS IoT Greengrass Core v2.3.0 sekarang tersedia.	7 Juli 2021
AWS kebijakan terkelola	Menambahkan informasi tentang kebijakan AWS terkelola untuk AWS IoT Greengrass.	2 Juli 2021
Opsi JVM baru yang direkomendasikan	Menambahkan informasi tentang opsi JVM yang direkomendasikan untuk mengontrol alokasi memori untuk AWS IoT Greengrass perangkat lunak Core.	30 Juni 2021
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.3.0	Rilis ini menyediakan versi 2.3.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Termasuk di dalamnya dukungan untuk dokumen konfigurasi komponen besar dalam deployment.	29 Juni 2021
Gambar Dockerfile dan Docker untuk AWS IoT Greengrass perangkat lunak Core v2.2.0	Gambar Dockerfile dan Docker untuk perangkat lunak AWS IoT Greengrass Core v2.2.0 sekarang tersedia.	28 Juni 2021

AWS IoT Device Tester v4.1.0 dengan GGV2Q v1.1.1 dirilis	Versi 4.1.0 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v1.1.1 dan mendukung penggunaan Greengras s nucleus v2.2.0, v2.1.0, dan v2.0.5 untuk kualifikasi perangkat.	18 Juni 2021
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.2.0	Rilis ini menyediakan versi 2.2.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Termasuk di dalamnya komponen yang dapat Anda deploy untuk menambahkan dukungan untuk perangkat klien dan menambahkan layanan bayangan lokal.	18 Juni 2021
Peluncur Lambda v2.0.6 dirilis	Komponen peluncur Lambda versi 2.0.6 tersedia. Versi ini mencakup beberapa peningkat an performa dan perbaikan bug.	13 Juni 2021
Komponen SageMaker Edge Manager baru dirilis	Versi 1.0.0 dari komponen Amazon SageMaker Edge Manager tersedia untuk. AWS IoT Greengrass Komponen ini menginstal biner agen SageMaker Edge Manager pada perangkat inti Greengras s.	10 Juni 2021

Jenis komponen	Menambahkan informasi tentang jenis komponen di AWS IoT Greengrass. Jenis komponen menentukan bagaimana Perangkat lunak inti AWS IoT Greengrass menjalankan komponen.	3 Juni 2021
AWS IoT Device Tester v4.0.2 dengan GGV2Q v1.1.0 dirilis	Versi 4.0.2 dari IDT untuk AWS IoT Greengrass V2 tersedia. Rilis ini mencakup suite kualifikasi AWS IoT Greengrass V2 (GGV2Q) v1.1.0 dan mendukung penggunaan Greengras s nucleus v2.1.0 dengan Greengrass CLI v2.1.0 untuk kualifikasi perangkat . Termasuk di dalamnya grup pengujian baru yang diperlukan untuk MQTT dan Lambda, dan perbaikan bug dan peningkatan kecil lainnya.	5 Mei 2021
Gambar Dockerfile dan Docker untuk AWS IoT Greengrass perangkat lunak Core v2.1.0	Gambar Dockerfile dan Docker untuk perangkat lunak AWS IoT Greengrass Core v2.1.0 sekarang tersedia. Gambar Docker memungkinkan Anda menjalankan perangkat lunak AWS IoT Greengrass Core dalam wadah Docker yang menggunakan Amazon Linux 2 sebagai sistem operasi dasar.	27 April 2021

AWS IoT Greengrass Pembaruan perangkat lunak Core v2.1.0	Rilis ini menyediakan versi 2.1.0 dari komponen inti Greengrass dan komponen yang disediakan pembaruan. AWS Ini mencakup komponen baru yang dapat Anda gunakan untuk mengunduh gambar Docker dari repositor i ECR Amazon pribadi, dan komponen sampel baru untuk melakukan inferensi pembelajaran mesin menggunakan Lite. TensorFlo w	26 April 2021
Contoh komponen yang menggunakan Secrets Manager	Menambahkan komponen contoh yang mencetak nilai AWS Secrets Manager rahasia yang Anda terapkan ke perangkat inti.	8 April 2021
AWS IoT Kebijakan minimal untuk perangkat inti Greengras s	Menambahkan informasi tentang serangkaian izin minimal yang diperlukan untuk mendukung fungsiona litas Greengrass dasar pada perangkat inti.	2 April 2021
Berlangganan aliran acara IPC	Menambahkan informasi tentang bagaimana menggunakan operasi interprocess communication (IPC) untuk berlangganan pengaliran peristiwa pada perangkat inti Greengrass.	1 April 2021

Support update untuk AWS IoT Device Tester for AWS IoT Greengrass	IDT untuk AWS IoT Greengrass V2 versi 4.0.1 sekarang mendukung penggunaan Greengrass nucleus versi 2.0.5 dengan Greengrass CLI versi 2.0.5 untuk kualifikasi perangkat.	17 Maret 2021
Buat komponen khusus yang menggunakan pengelola aliran	Menambahkan informasi tentang cara mengonfigurasi resep komponen dan artefak untuk mengembangkan aplikasi yang mengelola aliran data.	9 Maret 2021
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.0.5	Rilis ini menyediakan versi 2.0.5 dari komponen inti Greengrass dan komponen yang disediakan pembaruan . AWS Ini memperbaiki masalah dengan dukungan proxy jaringan dan masalah dengan titik akhir pesawat data Greengrass di Wilayah China. AWS	9 Maret 2021
Referensi variabel lingkungan komponen	Menambahkan informasi tentang variabel lingkungan yang ditetapkan perangkat lunak AWS IoT Greengrass Core untuk komponen. Anda dapat menggunakan variabel lingkungan ini untuk mendapatkan nama benda, Wilayah AWS, dan versi inti Greengrass.	23 Februari 2021

Instalasi manual	Menambahkan informasi tentang cara membuat AWS sumber daya yang diperlukan secara manual atau menginstal di belakang firewall atau proxy jaringan. Dengan menggunakan instalasi manual, Anda tidak perlu memberikan izin installer untuk membuat sumber daya di Anda Akun AWS, karena Anda membuat sumber daya yang diperlukan AWS IoT dan IAM. Anda juga dapat mengonfigurasi perangkat untuk tersambung pada port 443 atau melalui proksi jaringan.	17 Februari 2021
AWS IoT Greengrass Pembaruan pustaka IPC inti AWS IoT Device SDK untuk Python v2	Versi 1.5.4 dari untuk AWS IoT Device SDK Python v2 menyederhanakan langkah-langkah yang diperlukan untuk terhubung ke layanan Core IPC. AWS IoT Greengrass	11 Februari 2021
Support update untuk AWS IoT Device Tester for AWS IoT Greengrass	IDT untuk AWS IoT Greengrass V2 versi 4.0.1 sekarang mendukung penggunaan Greengrass nucleus versi 2.0.4 dengan Greengrass CLI versi 2.0.4 untuk kualifikasi perangkat.	5 Februari 2021

Tutorial baru untuk mengimpor fungsi Lambda	Menambahkan tutorial berbasis konsol baru untuk mengimpor fungsi Lambda sebagai komponen yang berjalan pada perangkat inti Greengrass.	5 Februari 2021
AWS IoT Greengrass Pembaruan perangkat lunak Core v2.0.4	Rilis ini menyediakan versi 2.0.4 dari komponen inti Greengrass. Termasuk di dalamnya parameter `greengrassDataPlanePort` yang baru untuk mengonfigurasi komunikasi HTTPS melalui port 443 dan memperbaiki bug. Kebijakan IAM minimal sekarang memerlukan `iam:GetPolicy` dan `sts:GetCallerIdentity` kapan penginstal perangkat lunak AWS IoT Greengrass Core dijalankan dengan. `--provision true`	4 Februari 2021
Komponen tunneling aman baru dirilis	Versi 1.0.0 dari komponen tunneling aman tersedia untuk. AWS IoT Greengrass Komponen AWS yang disediakan ini menggunakan tunneling AWS IoT aman untuk membangun komunikasi dua arah yang aman dengan perangkat inti Greengrass yang berada di belakang firewall terbatas.	21 Januari 2021

AWS IoT Device Tester untuk AWS IoT Greengrass v4.0.1 dirilis	Versi 4.0.1 dari IDT untuk AWS IoT Greengrass V2 tersedia. Versi ini memungkinkan Anda untuk menggunakan IDT untuk mengembangkan dan menjalankan serangkaian tes kustom Anda untuk validasi perangkat. Termasuk di dalamnya aplikasi IDT yang ditandatangani kode untuk macOS dan Windows.	22 Desember 2020
Rilis awal AWS IoT Greengrass Version 2	AWS IoT Greengrass V2 adalah rilis versi utama baru dari AWS IoT Greengrass. Versi ini menambahkan beberapa fitur seperti komponen perangkat lunak modular dan deployment berkelanjutan. Fitur-fitur ini memudahkan Anda untuk mengembangkan dan mengelola aplikasi edge.	15 Desember 2020

AWS Glosarium

Untuk AWS terminologi terbaru, lihat AWS glosarium di Referensi.Glosarium AWS

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